(1)

ARI Research Note 94-07

Software User's Manual for the Special Forces Military Occupational Specialties Allocation System

Fu Li, Khoi Do, and Ambrose Goicoechea
STATCOM, Inc.

for

Contracting Officer's Representative
Abraham Nelson

Leadership and Organizational Change Technical Area
Paul A. Gade, Chief

Manpower and Personnel Research Division
Zita M. Simutis, Director

79P

October 1993

United States Army
Research Institute for the Behavioral and Social Sciences

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

Research accomplished under contract for the Department of the Army

STATCOM, Inc.

Technical review by

Abraham Nelson

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 1993, October	3. REPORT TYPE AND DATES COVERED Final Aug 92 - Apr 93

4. TITLE AND SUBTITLE Software User's Manual for the Special Forces Military Occupational Specialties Allocation System	5. FUNDING NUMBERS MDA903-91-D-0024 63007A 792 2214 C05 T.O. 92-013
6. AUTHOR(S) Li, Fu; Do, Khoi; and Goicoechea, Ambrose	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) STATCOM, Inc. 7921 Jones Branch Drive, Suite 445 McLean, VA 22102	8. PERFORMING ORGANIZATION REPORT NUMBER --
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Institute for the Behavioral and Social Sciences ATTN: PERI-RP 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARI Research Note 94-07

11. SUPPLEMENTARY NOTES

--

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE --

13. ABSTRACT *(Maximum 200 words)*

This document describes the Special Forces (SF) Military Occupational Specialties (MOS) Allocation System and provides instructions for using and updating it. This computer software system enables a novice computer user to utilize the SF MOS Allocation model designed to aid in the allocation of soldiers to SF qualification courses. The objective of the model is to make assignments that maximize the probability of soldiers passing the course and, at the same time, to satisfy organizational constraints.

14. SUBJECT TERMS MOS allocation / Linear programming Special forces / Assignment model Personnel planning	15. NUMBER OF PAGES 78
	16. PRICE CODE --

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std Z39-18
298-102

SOFTWARE USER'S MANUAL FOR THE SPECIAL FORCES MILITARY OCCUPATIONAL SPECIALTIES ALLOCATION SYSTEM

CONTENTS

SOFTWARE USER'S MANUAL FOR THE SPECIAL FORCES MILITARY OCCUPATIONAL SPECIALTIES ALLOCATION SYSTEM

Section 1.0

GENERAL INFORMATION

The Manpower and Personnel Research Division, U.S. Army Research Institute for the Behavioral and Social Sciences (ARI), has a requirement to develop a user-friendly personal computer-based Special Forces (SF) Military Occupational Specialties (MOS) allocation model. This model will help improve the SF MOS classification system. This work is part of a larger ARI research program designed to assist the Special Forces in identifying, attracting, selecting, classifying, and retaining high quality soldiers.

The purpose of this project is to improve the efficiency of the SF MOS classification system. To achieve this goal an SF MOS allocation model is developed. This model will match soldiers who have successfully completed the Special Forces Assessment and Selection program to SF MOS in an attempt to reduce the number of MOS qualification course training failures and reclassification and, ultimately, to improve MOS performance.

This system is composed of computer software that enables individuals with limited computer skills to use it. The software integrates a personal computer user-friendly menu-driven system with a software package designed to solve mixed-integer mathematical programming problems.

1.1 PURPOSE OF THE SOFTWARE USER'S MANUAL

The purpose of this Software User's Manual is to provide the user with the information and step-by-step instructions to run the system. This system involves extracting data from the SF recruiter data base and soliciting information from the user. The SFMOS.EXE and LINDO software generates input, finds a solution, and displays that solution. This manual provides detailed instructions for preparing input files for the SFMOS.EXE program, model description, and a description of output results.

1.2 CONTENTS OF THE MANUAL

This manual is organized into four sections and three appendixes:

Section 1.0 provides general information on the SFMOS project, the purpose of the Software User's Manual, and the background information on the research effort. Contractor support for this research effort is briefly reviewed, as well.

Section 2.0, which describes the required software and hardware components and summarizes the system's functions, provides an overview of the SFMOS allocation system.

Section 3.0 presents detailed instructions for running the SFMOS.EXE program and discusses error recovery methods that can be used to modify the code and accommodate minor format changes in the input files, which may be needed over time.

Section 4.0 describes the allocation model and the format of the input and the output files used by LINDO.

Appendix A presents the SFMOS.EXE source code and includes descriptions of each function and procedure used in the program.

Appendix B presents a portion of the input and output files for the SFMOS.EXE program. These files are the input file for LINDO that is generated by SFMOS.EXE, the output file from LINDO, and the result file produced by the SFMOS.EXE program.

1.3 BACKGROUND

Dr. Abraham Nelson, U.S. Army Research Institute for the Behavioral and Social Sciences, is the researcher and sponsor of the SF MOS Allocation Project.

Dr. Ambrose Goicoechea, STATCOM, Inc., was the Project Manager responsible for concept design, system implementation, graphics production, and documentation.

Mr. Raymond Chin, ARI Information Systems Center, served as the government representative with responsibility for overseeing the project funding process. He was instrumental in the coordination of project management and research direction efforts by facilitating meetings attended by Dr. Abraham Nelson and Dr. Ambrose Goicoechea.

The SF MOS Allocation Project was developed during the period of August 1, 1992 to February 28, 1993.

1.4 APPROACH

In order to accomplish the goal of the project, the system is designed as a menu driven decision support system with access to the mixed-integer mathematical programming package LINDO, which supports the integer programming part of the system.

The major operations of the system are:

1. Read records containing a subset of the variables from an extract of the SF recruiter database of soldiers to be assigned to qualification courses (one for each SF MOS).

2. Solicit information from the user.

3. Generate input for LINDO and execute LINDO.

4. Determine an optimal allocation of the soldiers to the SF MOS qualification courses using LINDO.

5. Present the results in a format selected by the user.

SFMOS.EXE generates, with minimum user input, the input file compatible with LINDO's requirement of a MPS file format. This format is the Linear Programming (LP) format commonly used in industry. The MPS file is a text file format that makes it easier to move an LP model from one type machine (i.e., 386, 486, etc.) to another or from one computer manufacturer to another.

Section 2.0

DESCRIPTION OF THE SYSTEM

This section presents an overview of the system. It describes the hardware and software components and summarizes the functions supported by the SFMOS system.

2.1 Hardware and Software Components

This subsection identifies the major hardware and software components of the system. The functions each software package performs are also described.

The hardware required in the system is IBM compatible PCs and sufficient hard disk working space. The software packages used in the system are Hyper LINDO and the Borland C++ 3.0 compiler.

The minimum memory requirement for Hy/per LINDO to load is:

1. a 386 or better PC (the 386¦DOS EXTENDER requires this)

2. 158,624 bytes of conventional memory (141,520 bytes for 386¦DOS EXTENDER code and data plus 17,104 bytes for buffering in the conventional memory area)

3. a working math co-processor or a built-in Floating Point Unit (FPU)

4. three Megabytes (Megs) or more extended memory - the amount will determine how many rows and columns of data can be input into LINDO. With 3Megs of Extended Memory, the maximum rows and columns are 2000 x 4000 with maximum non-zeroes of 64,000. This limits the maximum number of soldiers that can be assigned to 400.

5. In order for LINDO to analyze data automatically when it is loaded, there must be a data file under the same LINDO directory called "INP_FILE.DAT." The user must make sure that either this directory is in the DOS path, or DOS "chdir" commands have been issued to where the data file is.

Usually these requirements are all met with 386 or better PCs.

Caution is necessary when loading expanded memory managers such as Quarterdeck's QEMM386.SYS or DOS/WINDOWS EMM386.EXE. The user should not pre-allocate Extended Memory by adding a switch in the loading statement "EXT=nnnn" in the CONFIG.SYS file.

The HYPER LINDO System is an interactive linear, quadratic, and integer programming system. Linear Programming (LP) is a mathematical procedure for determining optimal allocation of scarce resources. The methods for formulating and solving problems with integrality requirements are called Integer Programming. For this project, integer programming method is used to determine the problem solution.

Borland C++ 3.0 is a professional optimizing compiler for C++ and C developers. C++ is an object-oriented programming (OOP) language, and allows the user to take advantage of OOP's advanced design methodology and labor-saving features. C++ 3.0 is the next step in the natural evolution of C. Also, it is portable, which allows the user to easily transfer application programs written in C++ from one system to another. The user can also use C++ for almost any programming task. SFMOS.CPP is an application program written in the Borland C++ 3.0.

If the user needs to work with the source code of the SFMOS.EXE program, please read the following instructions:

* Get into C:\TC\BGI\> directory (TC means TURBO C++)

```
*    Type BGIOBJ /F SANS       (to create SANSF.OBJ file)
          BGIOBJ /F LITT       (to create LITTF.OBJ file)
          BGIOBJ /F GOTH       (to create GOTHF.OBJ file)
          BGIOBJ /F TRIP       (to create TRIPF.OBJ file)
          BGIOBJ /F EGAVGA     (to create EGAVGAF.OBJ file)

*    Copy all of the OBJ files to C:\TC\LIB\>

*    Change the directory to C:\TC\LIB\>

*    Type
       TLIB GRAPHICS +EGAVGAF +SANSF +LITTF +GOTHF +TRIPF

*    Copy EGAVGA.BGI to the same directory of SFMOS.CPP
loaded.

*    Copy SFMOS.H to the directory C:\TC\INCLUDE (or INC).
```

2.2 Summary of Functions Supported by SF MOS System

There are five main functions to support users of the SF MOS system. These functions are listed as the system Main Menu:

Main Menu

1. Create Parameter Data File
2. Modify Parameter Data File
3. Run MOS-ARI Model
4. Display Run Results
5. Print Report of Results

Function 1 must be selected by a user when using the system for the first time. It solicits information that is used in the generation of input to LINDO. Function 1 includes preset default values that are used in the absence of user input.

Function 2 allows a user to update parameters in order to run the system again. This allows the user to examine alternative solutions.

Function 3 generates the input to LINDO and runs the LINDO software package that produces a solution to the problem.

Function 4 displays the results in a format based on the user's selection from various options.

Function 5 outputs the results to an ASCII file based on the user's specification. This enables the user to get a hard copy printout of the results.

On the bottom of each screen, the function of the 6 function keys is indicated:

F1 to continue the process,
F2 to show the help screen/window (some of the help screens require user inputs),
F3 to save the current change,
F4 to go to the previous screen,
F5 to go to the main menu, and
F6 to exit the system to DOS.

Following the selection of the function in the main menu by a user, the system displays sub-menus on the screen. These sub-menus are described as follows:

Function 1, "Create Parameter Data File," displays the total number of soldiers to be assigned to the four SF qualification courses. At this point the user must input the desired number of soldiers to be assigned to the Special Operation Weapon Sergeant (18B), Special Operation Engineer Sergeant (18C), Special Operation Medical Sergeant (18D), and Special Operation Communication Sergeant (18E) courses. After the user inputs numbers for each course, the system will sum them to determine if they are equal to the total number of soldiers to be assigned. If they are not equal, a warning message will be shown on the screen and the user must change the numbers so that they add up to the total number of soldiers to be assigned.

Next, the user can change the Armed Services Vocational Aptitude Battery (ASVAB) composite cut-scores for each course. The default score for each cut-score is 80. The subsequent option the user has is to change the grade restriction for each course. The default grades are E-5 for 18B and E-4 for the other three courses. The user can also input MOS preference information on meeting the goal for the desired number of soldiers in each course. The possible inputs are numbers from 1 to 10 where 1 is the lowest value and 10 the highest.

Function 2, "Modify Parameter Data File," displays are the same as Function 1 except all previous input data are shown. The user can make changes to these data and to rerun the system.

Function 3, "Run MOS-ARI Model," will tell the user which process the system is running.

Function 4, "Display Run Results," displays the following sub-menu:

Result Menu

1. Summary Of The Result
2. Individual Record By SSN
3. Individual Record By MOS
4. Individual Record By WANT.

Selection 1 shows the course title (Course), the number of soldiers assigned to each course (Soldiers), the number of soldiers who wanted the course, and the number of soldiers who were assigned the course they wanted (Match), and the average scores of CO, EL, FA, GT, SC, and ST for each course.

Selection 2 shows the social security number (SSN) for each soldier, the course title that the soldier wanted (WANT), the course title to which that soldier was assigned (MOS), grade, and the six ASVAB composite scores noted above.

Selection 3 and selection 4 show the same information described in selection 2 for each soldier based on the different sorting orders. (Note that if a soldier is not qualified for any courses his record will show as the top record by MOS selection and the value for MOS is "000".)

Function 5, "Print Report of Results," displays the same sub-menu as in Function 4, "Display Run Results." Instead of displaying these results on the screen the system copies them to ASCII files the user can print them out. The file name corresponding to the selection is as follows:

Selection	File name
1. Summary Of The Result	SFMOSRPT.DAT
2. Individual Record By SSN	BYSSNRPT.DAT
3. Individual Record By MOS	BYMOSRPT.DAT
4. Individual Record By WANT.	BYWANRPT.DAT

Section 3.0

INSTRUCTIONS FOR RUNNING SFMOS.EXE PROGRAM

To run the SFMOS.EXE program, the user needs to thoroughly understand the model used in the system. This is necessary so that the required input file can be properly built. The model is presented in Section 4.0.

3.1 Step 1: Preparing the Input Files for SFMOS.EXE

Before running the SFMOS.EXE program, two input files are needed in the same directory with SFMOS.EXE. These files are described as follows: file name, format, and sample data. Note that the first file is created using a dBASE program. The second file can be created by a DOS editor or any available editor on the PC.

1. SFMOS.DAT -- This is an ASCII extract of the SF recruiter database that is a dBASE file. It includes all soldiers information required by the system. The format of this data file should follow the format below:

Field	Field Name	Type	Width
1	SFASCLASS	Character	6
2	SFAS_1ST	Character	6
3	SFAS_2ND	Character	6
4	SFAS_3RD	Character	6
5	SFAS_4TH	Character	6
6	SSN	Character	11
7	WANT	Character	3
8	MOS	Character	3
9	GRADE	Character	3
10	CO	Numeric	3
11	EL	Numeric	3
12	FA	Numeric	3
13	GT	Numeric	3
14	SC	Numeric	3
15	ST	Numeric	3
16	APT	Numeric	3
17	DLAB	Logical	1
18	SCORE	Numeric	3

2. C_ARRAY.DAT -- This data file includes 6 rows and 4 columns. The data in each row are assigned to an array during the system process. They are as shown below:

```
C = ( .047, .363, .247, .123)
CONSTANT = (-3.0606, -4.4982, -.40038, -1.86797)
COEFAA1 = (.2614, .40144, .2905, .19524)
COEFAA2 = (0.0, 0.0, 0.0, 0.0)
COEFMO = (.2344, .40144, .5343, 0.0)
COEFW = (0.0, 0.0, 0.0, 0.0)
```

Note that these data are notional.

3.2 Step 2: Running the SFMOS.EXE Program

First, the user needs to copy all files to the same directory on one of the hard drives. Go to that directory and type "SFMOS" to start the system. The program should start with the first screen: "WELCOME to MOS Special Forces Allocation Model."

After the program is started, simply follow the menu and select the desired function. Usually, the user will select Function 1, "Create Parameter Data File," in order to input the data that the system needs. These data will be saved as data files called MODIFYS1.DAT and MODIFYS2.DAT, one for each screen.

Function 2, "Modify Parameter Data File," allows modifications of the above files.

Function 3, "Run MOS-ARI Model," allows the system to create the input file for LINDO. This file is named INP_FILE.DAT. Then, the system starts the LINDO package. The output from LINDO is placed in the file named OUT_FILE.DAT.

After LINDO finds an optimal solution, the system reorganizes the results from the output file into a database file that the user can read more easily.

Function 4 displays the results in a format based on user selected options.

Function 5 writes the results to ASCII files to give the user option of printing them.

The user may wish to rerun the system. Function 2 provides the user with access to those data that were previously input to the system. Here, the user can make any changes and run the system again.

Intermediate files can be found in the same directory with the names as defined above.

Section 4.0

DESCRIPTION OF THE SPECIAL FORCES MOS ALLOCATION MODEL

Decision Variables

The decision variables used in this model are defined as follows:

X_{ij} = the assignment of the ith soldier to the jth MOS,
D^+_j = the deviation above the goal, the desired number of soldiers, for MOS j, and
D^-_j = the deviation below the goal, the desired number of soldiers, for MOS j.

Constants

The constants used in this model are defined as follows:

P_{ij} = the value of assigning the ith soldier to the jth MOS,
$GOAL_j$ = the goal, the desired number of soldiers, for the jth MOS,
W^+_j = the penalty (weight) for deviating above the goal for MOS j,
W^-_j = the penalty (weight) for deviating below the goal for MOS j.
N = the total number of soldiers to be assigned,
SC_j = the cut-score on the ASVAB composite required for MOS_j, and
AA_{ij} = the ith individual's ASVAB composite associated with the MOS_j.

The objective of the model is to:

$$MAXIMIZE \sum_{i=1}^{N} \sum_{j=1}^{4} P_{ij} * X_{ij} + \sum_{j=1}^{4} W_j^+ * D_j^+ + \sum_{j=1}^{4} W_j^- * D_j^-$$

Subject to:

Each soldier is assigned to one and only one MOS.

$$\sum_{j \in \{j | A_{ij} \geq S_j\}} X_{ij} = 1 \quad \forall i$$

All soldiers are assigned.

$$\sum_{i=1}^{n} \sum_{j=1}^{4} X_{ij} = N$$

Goal Constraints

$$\sum_{i=1}^{n} X_{ij} - D_j^+ + D_j^- = GOAL_j \quad \forall j$$

Integer Value Constraints

$$X_{ij} = 0,1 \quad \forall i,j$$

Problem Size

The dimensions of the problem are as follows:

the number of variables (columns) = 4*(N + 2),
the number of constraints (rows) = N + 6 (This includes the objective function), and
the number of integer variables = N*4.

Determination of the Values for the Objective Function

Let AA = {CO, FA, GT, ST, EL, SC} = ASVAB composite scores.

Steps for estimating the values of the objective function are listed as follows:

Functions for assignment to 18B:

$$f(\overline{x}) = \frac{1}{1 + e^{-CONSTANT(1) - COEFAA1(1)\cdot CO - COEFAA2(1)\cdot 0 - COEFMO(1)\cdot COMBAT - COEFW(1)\cdot WANTB}}$$

Function for assignment to 18C:

$$f(\overline{x}) = \frac{1}{1 + e^{-CONSTANT(2) - COEFAA1(2)\cdot FA - COEFAA2(2)\cdot CO - COEFMO(2)\cdot COMBAT - COEFW(2)\cdot WANTC}}$$

Function for assignment to 18D:

$$f(\overline{x}) = \frac{1}{1 + e^{-CONSTANT(3) - COEFAA1(3)\cdot ST - COEFAA2(3)\cdot GT - COEFMO(3)\cdot HEALTH - COEFW(3)\cdot WANTD}}$$

Function for assignment to 18E:

$$f(\overline{x}) = \frac{1}{1 + e^{-CONSTANT(4) - COEFAA1(4)\cdot EL - COEFAA2(4)\cdot SC - COEFW(4)\cdot WANTE}}$$

where $\overline{x}$ is a vector and

C = (.047, .363, .247, .123),
CONSTANT = (-3.0606, -4.4982, -.40038, -1.86797),
COEFAA1 = (.2614, .40144, .2905, .19524),
COEFAA2 = (0.0, 0.0, 0.0, 0.0),
COEFMO = (.2344, .40144, .5343, 0.0), and
COEFW = (0.0, 0.0, 0.0, 0.0).

The numbers in the arrays are the elements of C_ARRAY.DAT. Note that these numbers are illustrative.

Definition of COMBAT and HEALTH:

PMOS is a three character variable in the data base. The first two characters indicate the Career Management Field (CMF) for a military occupational specialty.

```
IF
          11 <= CMF <= 19           THEN COMBAT = 1
                                    ELSE COMBAT = 0.
IF
          CMF = 91 OR CMF = 92      THEN HEALTH = 1
                                    ELSE HEALTH = 0.
```

Definition of WANTB, WANTC, WANTD, and WANTE:

WANT is a variable on the data base.

```
IF WANT = 18B  THEN WANTB = 1
               ELSE WANTB = 0.

IF WANT = 18C  THEN WANTC = 1
               ELSE WANTC = 0.

IF WANT = 18D  THEN WANTD = 1
               ELSE WANTD = 0.

IF WANT = 18E  THEN WANTE = 1
               ELSE WANTE = 0.
```

APPENDIX A. SFMOS.EXE SOURCE CODE

```
/****************************************************************
********
     SFMOS.CPP
*****************************************************************
********/
#include <dos.h>
#include <dir.h>
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <mem.h>
#include <alloc.h>
#include <string.h>
#include <math.h>
#include <io.h>
#include <sys\stat.h>
#include <sys\types.h>
#include <fcntl.h>
#include <process.h>
#include <sfmos.h>

int Set_Graph(void)
/****************************************************************
*****/
/*   This  function  initializes  the  graphic  system  and  register
driver */
/*  and font that was added into graphics.lib.  Your monitor must
be */
/*  EGA or VGA to work with this program.
  */
/*  This function is called by Main ().
  */
/****************************************************************
*****/
   {
   int graphdriver = DETECT, graphmode, error_code;
   /* Register a driver that was added into graphics.lib */
   error_code = registerfarbgidriver (EGAVGA_driver_far);
   if (error_code < 0)
      {
      printf ("Graphics error: %s\n", grapherrormsg(error_code));
      printf ("Press any key to halt: ");
      getch ();
      return (-1);
      }
   /* Register triplex font files that was added into graphics.lib
*/
   error_code = registerfarbgifont (triplex_font_far);
   if (error_code < 0)
      {
      printf ("Graphics error: %s\n", grapherrormsg(error_code));
      printf ("Press any key to halt: ");
      getch ();
```

```
        return (-1);
        }
     /* Register small font files that was added into graphics.lib */
     error_code = registerfarbgifont (small_font_far);
     if (error_code < 0)
        {
        printf ("Graphics error: %s\n", grapherrormsg(error_code));
        printf ("Press any key to halt: ");
        getch ();
        return (-1);
        }
     /*  Register  sansserif  font  files  that  was  added  into
  graphics.lib */
     error_code = registerfarbgifont (sansserif_font_far);
     if (error_code < 0)
        {
        printf ("Graphics error: %s\n", grapherrormsg(error_code));
        printf ("Press any key to halt: ");
        getch ();
        return (-1);
        }
     /* Register gothic font files that was added into graphics.lib
  */
     error_code = registerfarbgifont (gothic_font_far);
     if (error_code < 0)
        {
        printf ("Graphics error: %s\n", grapherrormsg(error_code));
        printf ("Press any key to halt: ");
        getch ();
        return (-1);
        }
     /* Initialize graphics system; must be EGA or VGA */
     initgraph(&graphdriver, &graphmode, "..\\bgi");
     error_code = graphresult();
     if (error_code != grOk)
        {
        printf ("No graphics hardware found");
        return (-1);            /* No graphics hardware found */
        }
     if ((graphdriver != EGA) && (graphdriver != VGA))
        {
        closegraph();
        printf ("No graphics EGA or VGA");
        return 0;
        }
     return (1);                /* Graphics is OK. Return true */
     }   /* end Set Graphics */

  void Drawborder(void)
  /*****************************************************************
  *******/
  /*  This function draws the border and set background color on the
  srn */
  /*  It is called by all of the functions which creates screen.
```

```
    */
/*****************************************************************
*******/
   {
   int x1,x2,y1,y2,h,l;
   int x,y;
   int i, colr,fs;
   clearviewport();
   setbkcolor(9);
   setlinestyle( 0,0,3 );
   getviewsettings( &vp );
   l = 10;
   x = vp.right;
   y = vp.bottom;
   for ( i=1; i<4; ++i)
      {
      switch(i)                        /* setup for border pattern */
      {
      case 1: h =  0; colr = 14; fs = 8; break;
      case 2: h += l; colr = 6; fs = 7; break;
      case 3: h += l; colr = 11; break;
      }
      setcolor(colr);
      rectangle( h, h, x-h,y-h);
      if (i<3)
      {
      setfillstyle(fs,colr);
      bar(h,h,x-h,h+l);       /*  draw row strip of margin  */
      bar(h,y-h-l,x-h,y-h);
      bar(h,h+l,h+l,y-h-l);  /*  draw column strip of margin */
      bar(x-h-l,h+l,x-h,y-h-l);
      }   /* if */
      }   /* for */
   }   /* drawborder */

int Get_Fn_Key (void)
/*****************************************************************
*******/
/*  This function get rid of the first ascii value of the function
key */
/*****************************************************************
*******/
   {
   int fkey_val;              /* funckey value */
   fkey_val = getch();       /* Read ascii value of keyboard pressed
*/
   if (fkey_val == NULL)    /* it is function key */
      {
      is_funckey = TRUE;
      fkey_val = getch();
      }
   else                       /* not a function key */
      is_funckey = FALSE;
   return (fkey_val);
   }   /* get function key */

void Functn_Bar(int distlett)

/*********************************************************************
******/
   /*  This function draws the six function bars at the botton of
every  */
   /*  menu and data entry screen.
      */
   /*  F1 for Continue, F2 for Help, F3 for Save, F4 for Previous
Screen */
   /*  F5 for Mainmenu, and F6 for Exit the program.
      */
   /*  Variable (int distlett) is the lenght of the variable bar.
      */
   /*  This function is called by  Graph_Main_Menu (),
      */
   /*                              Create_Scr1 (),
      */
   /*                              Create_Scr2 ().
      */

/*********************************************************************
******/
   {
   const distbord = 25;    /* distance from border to screen */
   int i, x, y, xmax, ymax, portion;
   char *funckeys[7], *funcmess[7];   /* funckeys F1 to F6 and
messages */
   /* initialize function keys and function messages */
   funckeys[1] = "   F1   "; funckeys[2] = "   F2   ";
   funckeys[3] = "   F3   "; funckeys[4] = "   F4   ";
   funckeys[5] = "   F5   "; funckeys[6] = "   F6   ";
   funcmess[1] = "Continue"; funcmess[2] = "  Help  ";
   funcmess[3] = "  Save  "; funcmess[4] = "Pre_Scrn";
   funcmess[5] = "MainMenu"; funcmess[6] = "  Exit  ";
   xmax = getmaxx()-distbord;      /* Get max right-bottom corner
position */
   ymax = getmaxy()-distbord;
   x = distbord;
   y = getmaxy()-distbord*3;
   setcolor (LIGHTGREEN);
   rectangle (x, y, xmax, ymax);                /* Draw a bar at the
bottom  */
   setfillstyle (SOLID_FILL, LIGHTGREEN);    /* of the screen and
fill it */
   floodfill (x+5, y+5, 10);                  /* with light green
color    */
   setcolor (BLUE);
   /* Draw a horizontal line cut the light green bar in half */
   line (x, y+distbord, xmax, ymax-distbord);
   portion = (xmax-distbord)/6;
   settextstyle (1, 0, 0);
   setusercharsize (1, 2, 2, 3);
   bar_x1[1] = x;
   bar_y1[1] = y;
   bar_x2[1] = x+portion;
   bar_y2[1] = ymax;
   for (i=1; i<=6; i++)       /* Draw five vertical lines cut the
light */
      {                        /* green bar into six even pieces.
      */
      if (i < 6)
      {
      line (x+i*portion, y, x+i*portion, ymax);
      bar_x1[i+1] = x+portion*i;
      bar_y1[i+1] = y;
      bar_x2[i+1] = x+portion*(i+1);
      bar_y2[i+1] = ymax;
      }
      /* put function name and message into six light green bar */
      outtextxy (x+(i-1)*portion+distlett, y, funckeys[i]);
      outtextxy (x+(i-1)*portion+distlett, y+25, funcmess[i]);
      }  /*  end for  */
   }  /* end function bar */

void Draw_Win (int x1, int y1, int x2, int y2, int color)

/*****************************************************************/
   /*  This function draws a bar from the x1, y1, x2, y2 position
 */
   /*  of that bar and color.
 */

/*****************************************************************/
   {
   setcolor (color);                                /* Set color
*/
   rectangle (x1, y1, x2, y2);               /* Draw a rectangle
*/
   setfillstyle (SOLID_FILL,color);         /* Set to solid fill
*/
   floodfill (x1+5,y1+5,color);             /* Fill the rectangle
*/
   setcolor (YELLOW);                  /* Set color back to yellow
*/
   }  /* end draw window */

void F6message()

/******************************************************************
**/
   /* this function displays the warning window with warning
message */
   /* whenever user press F6 to exit or Mouse click on F6 function.
*/
   /* The warning message asks for saving the data entered or not.
 */
   /* This function is called by  Graph_Main_Menu (),
  */
   /*                              Create_Scr1 (),
  */
   /*                              Create_Scr2 ().
  */

/*********************************************************************
**/
   {
   void *rect;            /* saves memory of F6 warning window to
RAM */
   int x1, x2, y1, y2;             /* x,y position of F6 warning
window */
   int f6flag;                        /* Set flag to leave while
loop */
   unsigned int size;        /* memory size of warning message
image */
   x1 = getmaxx () - 225;               /* set x,y exit window
size */
   y1 = getmaxy () - 150;
   x2 = getmaxx () - 25;
   y2 = getmaxy () - 80;
   func_key = FALSE;
   size = imagesize (x1, y1, x2, y2);      /* get memory size of
image */
   rect = malloc (size);
   getimage  (x1, y1, x2, y2, rect);
   settextstyle (2,0,6);
   setlinestyle (0,0,2);
   settextjustify (LEFT_TEXT, TOP_TEXT);        /* set text to the
left */
   Draw_Win  (x1, y1, x2, y2, LIGHTCYAN);
   setcolor  (LIGHTMAGENTA);
   rectangle (x1+5,y1+5,x2-5,y2-5);                  /* draws two
rectangles  */
   rectangle (x1+7,y1+7,x2-7,y2-7);              /* inside window
message */
   setcolor  (LIGHTCYAN);
   line  (x1+40,y1+5,x2-40,y2-(y2-y1)+5); /* Draw two lines with
reverse */
   line  (x1+40,y1+7,x2-40,y2-(y2-y1)+7); /* color to put 'Exit
Program' */
   setcolor  (LIGHTMAGENTA);
   outtextxy (x1+32, y1-2, " Exit Program ");/* Bold face the word
Exit */
   outtextxy (x1+33, y1-2, " Exit Program ");
   outtextxy (x1+32, y1+15, " Save Change? ");/* Bold face Save
Change? */
   outtextxy (x1+33, y1+15, " Save Change? ");
   setlinestyle (0,0,3);
   Draw_Win  (x1+18,y1+41,x1+58,y1+58, DARKGRAY);
                                        /* Shadow for yes window */
   Draw_Win   (x1+15,y1+38,x1+55,y1+55, LIGHTMAGENTA);
                                           /* Yes window */
   setcolor   (BLUE);
   outtextxy (x1+22,y1+38, "Y");                /* Write 'Y' to Yes
window */
   outtextxy (x1+23,y1+38, "Y");              /* Bold face the word
'Y' */
   setcolor   (YELLOW);
   outtextxy (x1+33,y1+38, "es");              /* Write 'es' to Yes
window */
   outtextxy (x1+34,y1+38, "es");             /* Bold face the word
'es' */
   Draw_Win   (x1+68,y1+41,x1+108,y1+58, DARKGRAY);
                                           /* Draw shadow for 'No' */
   Draw_Win   (x1+65,y1+38,x1+105,y1+55, LIGHTMAGENTA); /* Draw No
window */
   setcolor   (BLUE);
   outtextxy (x1+77,y1+38, "N");                                   /*
Draw 'N' */
   outtextxy (x1+78,y1+38, "N");                              /* Bold
face 'N' */
   setcolor   (YELLOW);
   outtextxy (x1+88,y1+38, "o");                                   /*
Draw 'o' */
   outtextxy (x1+89,y1+38, "o");                              /* Bold
face 'o' */
   Draw_Win   (x1+118,y1+41,x1+188,y1+58, DARKGRAY);
                                           /* Draw shadow for Cnl */
   Draw_Win   (x1+115,y1+38,x1+185,y1+55, LIGHTMAGENTA);
                                           /* Draw Cancel Win */
   setcolor   (BLUE);
   outtextxy (x1+124,y1+38, "C");                                  /*
Draw 'C' */
   outtextxy (x1+125,y1+38, "C");                             /* Bold
face 'C' */
   setcolor   (YELLOW);
   outtextxy (x1+135,y1+38, "ancel");                         /* Draw
'ancel' */
   outtextxy (x1+136,y1+38, "ancel");                    /* Bold face
'ancel' */
   setlinestyle (0,0,2);
   settextjustify (CENTER_TEXT, TOP_TEXT);
     /* while y,n,c is not pressed or mouse is not clicked on the
three */
     /* rectangle 'yes','no','cancel' then wait for user action.
     */
   f6flag = FALSE;
   while (f6flag == FALSE)
      {
      func_key = Get_Fn_Key();
      f6flag = TRUE;
      switch (func_key)
      {
      case 121: ;
```

```
        case 89 :                           /* Letter Y or y is pressed
*/
           func_key = TRUE;
           break;
        case 110: ;
        case 78 :                           /* Letter N or n is pressed
*/
           func_key = 2;
           break;
        case 99 : ;
        case 67 :                           /* Letter C or c is pressed
*/
           func_key = FALSE;
           break;
        default :
           f6flag = FALSE;
           break;
        }  /* end switch */
        }  /* end while f6flag = false */
     putimage (x1,y1,rect,COPY_PUT);     /* put back the background
image */
     free (rect);                            /* free memory of image
from RAM */
     }  /* end F6 message */

int Graph_Title(void)
   /*****************************************************************/
   /*  This function draws the title of the MOS Special Forces.  */
   /*  This function is calling  Drawborder ()                    */
   /*            and is called by Main ().                        */
   /*****************************************************************/
   {
   int h,y,xcenter;
   int func_key = 0;
   Drawborder();
   xcenter = (vp.right - vp.left)/2;
   setcolor(12);
   settextstyle(1,0,6);
   h = textheight("H");
   y = 0.5*h;
   settextjustify (CENTER_TEXT, TOP_TEXT);
   outtextxy(xcenter,y,"WELCOME");
   y += h;
   outtextxy(xcenter,y,"to");
   y += 1.5*h;
   setcolor(YELLOW);
   outtextxy(xcenter,y,"MOS Special Forces");
   y += h;
   outtextxy(xcenter,y,"Allocation Model");
   setcolor (LIGHTCYAN);
   settextstyle(1,0,4);
   h = textheight("H");
   y += 3*h;
   outtextxy(xcenter,y,"by");
   y += 1.5*h;
   outtextxy(xcenter,y,"U.S.Army Research Institute (ARI)");
   setcolor (YELLOW);
   y += 2*h;
   Draw_Win(223,413,313,443,DARKGRAY);
   Draw_Win(220,410,310,440,MAGENTA);
   outtextxy(xcenter,y,"Press Enter to continue");
   /* F1, F6, Enter, or MouseKey is pressed */
   while ((func_key != 59) && (func_key != 64) &&
       (func_key != 13)) func_key = Get_Fn_Key();
   return (func_key);
   }   /* end graph title */

int Graph_Intro(void)

/*****************************************************************
*****/
   /*  This function draws an introduction  of the MOS Special
Forces.  */
   /*  This function is calling   Drawborder ()
     */
   /*              and is called by Main ().
     */

/*****************************************************************
*****/
   {
   int i,h,y,xcenter,func_key = 0;
   char *intritem [10];
   Drawborder();
   xcenter = (vp.right - vp.left)/2;
   for (i=1; i<=7; i++)
      intritem [i] = "                                    ";
   intritem [1] = "  MOS-ARI is a model that allocates ";
   intritem [2] = "  soldiers who have completed the   ";
   intritem [3] = " Special Forces (SF) Assessment and ";
   intritem [4] = " Selection program to qualification ";
   intritem [5] = " courses for the SF MOS.            ";
   intritem [6] = "Press Enter to continue";
   setcolor(LIGHTRED);
   settextstyle(1,0,6);
   h = textheight("H");
   y = 0.5*h;
   settextjustify (CENTER_TEXT, TOP_TEXT);
   outtextxy(xcenter,y,"INTRODUCTION");
   setcolor (LIGHTCYAN);
   settextstyle(1,0,4);
   h = textheight("H");
   y += 3*h;
   for (i=1; i<=5; i++)
      {
      outtextxy (xcenter,y,intritem [i]);
      y += 1.5*h;
      }   /* end for */
   y += 1.5*h;
   Draw_Win (223,403,313,433,DARKGRAY);
   Draw_Win (220,400,310,430,MAGENTA);
   setcolor (YELLOW);
   outtextxy (xcenter,y,intritem [6]);
   /* F1, F4, F6, Enter, or MouseKey is pressed */
   while ((func_key != 59) && (func_key != 64) && (func_key != 62)
&&
         (func_key != 13)) func_key = Get_Fn_Key();
   return (func_key);
   }   /* end Graph Introduction */

void Drawhelp (int x1, int y1, int x2, int y2)

/****************************************************************
*****/
   /*  This function draws a rectangle and fills it with lightcyan
then */
   /*  draws another two rectangle with different color and the
title   */
   /*  Help.  This function is calling Draw_Win ()
      */
   /*                    and is called by Help_Win ().
      */

/****************************************************************
*****/
   {
   int i,j,cntr;                                        /* temporary
variables */
   i = x2;
   j = y2;
      /* fill up the x1, y1, x2, y2 rectangle from right, bottom
up  */
   for (cntr=1; cntr<=20; cntr++)
      {
      i -= 20;
      j -= 10;
      Draw_Win (i, j, x2, y2, LIGHTCYAN);                   /* Paint
window */
      }
   setcolor (LIGHTMAGENTA);
   rectangle  (x1+5,  y1+5,  x2-5,  y2-5);          /* draws  two
rectangles  */
   rectangle  (x1+7,  y1+7,  x2-7,  y2-7);        /* inside window
message */
   setcolor (LIGHTCYAN);
   line  (x1+170,  y1+5,  x2-173,  y2-(y2-y1)+5);       /* draws two
lines */
   line  (x1+170,  y1+7,  x2-173,  y2-(y2-y1)+7);    /* with inverse
color */
   setcolor (LIGHTMAGENTA);
   outtextxy (x1+201, y1-5, " Help ");      /* bold face the word
Help */
```

```
   outtextxy (x1+200, y1-5, " Help ");
   }   /* Draw Help */

void Help_Win (int messno)

/*******************************************************************
****/
   /*  This function displays the Help window fill with help
messages  */
   /*  and also create functions for Up, Down, PageUp, and PageDown
  */
   /*  keys.  It works in any DataEntry screen whenever user
pressed  */
   /*  function key F2 and F7 is for close Help and back to entry
scrn.*/
   /*  This function is calling  Drawhelp (),    Draw_Win ()
    */
   /*          and is called by  Create_Scr1 (),  Create_Scr2 ().
    */

/*******************************************************************
****/
   {
   void *rect;          /* Save memory of help window image size to
ram */
   int maxarr, i;                /* Max array, function key, temp
variable */
   int firstln, last_ln;   /* First line and lastline appear in
window */
   int x1, x2, y1, y2, cntr;              /* x,y position of help
window */
   unsigned int size;                   /* Memory size of message
images */
   char *mess1[31], *mess2[31];                 /* Help messages 1
and 2 */
   func_key = 0;                                     /* Initialize
variables */
   firstln  = 1;
   last_ln  = 10;
   settextjustify (CENTER_TEXT, TOP_TEXT);   /* Set text to the
middle */
   x1 = (getmaxx()/2)-200;        /* Get x,y's position of help
window */
   x2 = x1+400;
   y1 = (getmaxy()/2)-100;
   y2 = y1+200;
   for (i=1; i<=30; i++)   /* Enter help message here */
      {
      mess1[i] = "                              ";
      if (i <= 10)
      mess1[i] = "The test is for create one page1";
      else
      if (i <= 20)
         mess1[i] = "The test is for create one page2";
```

```
        else
           mess1[i] = "The test is for create one page3";
        mess2[i] = "                                    ";
        if (i <= 10)
        mess2[i] = "The test is for create two page1";
        else
        if (i <= 20)
           mess2[i] = "The test is for create two page2";
        else
           mess2[i] = "The test is for create two page3";
        }   /* end for */
    setlinestyle (0,0,2);
    size = imagesize (x1, y1, x2, y2);       /* get memory size of
image */
    rect = malloc (size);
    getimage (x1, y1, x2, y2, rect);              /* Get background
image */
    settextstyle (2,0,6);
    Drawhelp (x1, y1, x2, y2);
    while (func_key != 65)                   /* Function key F7 is
pressed */
        {
        cntr = 0;
        switch (messno)
        {
        case 1:                   /* Write help message 1 into window
*/
           for (i=firstln; i<=last_ln; i++)
              {
              ++cntr;
              outtextxy (x1+200, y1+(cntr*15), mess1[i]);
              }
           maxarr = 30;
           break;
        case 2:                   /* Write help message 2 into window
*/
           for (i=firstln; i<=last_ln; i++)
              {
              ++cntr;
              outtextxy (x1+200, y1+(cntr*15), mess2[i]);
              }
           maxarr = 30;
           break;
        }   /* end switch messno */
        Draw_Win (273,313,313,331,DARKGRAY);                /* paint
window */
        Draw_Win (270,310,310,328,LIGHTMAGENTA);
        setcolor (BLUE);
        outtextxy (x1+201, y2-30, "Press  F7  to end Help");
        outtextxy (x1+200, y2-30, "Press  F7  to end Help");
        setcolor (LIGHTCYAN);
        cntr = 0;
        func_key = Get_Fn_Key();
        while ((func_key != 65) && (func_key != 72) &&
            (func_key != 73) && (func_key != 80) && (func_key != 81))
        func_key = Get_Fn_Key();
        switch (messno)
        {
        case 1:                                 /* erase old help message
1 */
           for (i=firstln; i<=last_ln; i++)
              {
              ++cntr;
              outtextxy (x1+200, y1+(cntr*15), mess1[i]);
              }
           break;
        case 2:                                 /* erase old help message
2 */
           for (i=firstln; i<=last_ln; i++)
              {
              ++cntr;
              outtextxy (x1+200, y1+(cntr*15), mess2[i]);
              }
           break;
        }  /* end switch messno */
        setcolor (LIGHTMAGENTA);
        switch (func_key)
        {
        case 72:                  /* Up key is pressed */
           if (firstln > 1)
              {
              --last_ln;
              --firstln;
              }  /* end if */
           break;
        case 80:                  /* Down key is pressed */
           if (last_ln < maxarr)
              {
              ++last_ln;
              ++firstln;
              }  /* end if */
           break;
        case 73:                  /* Page Up key is pressed */
           if (firstln <= 10)
              {
              firstln = 1;
              last_ln = 10;
              }
           else
              {
              firstln -= 10;
              last_ln -= 10;
              }
           break;
        case 81:                  /* Page Down key is pressed */
           if (last_ln <= maxarr-9)
              {
              firstln += 10;
```

```
               last_ln += 10;
               }
            else
               {
               firstln = maxarr - 9;
               last_ln = maxarr;
               }
            break;
        }   /* end switch func_key */
        }   /* end while function key */
   putimage (x1, y1, rect, COPY_PUT);  /* Put back the background
image */
   free (rect);                       /* free memory of image from
RAM */
   setlinestyle (0,0,3);
   func_key = 0;
   }   /* end Help window */

void Bar_Mess (char *err_message)

/*****************************************************************
*****/
   /*  This function draws a message bar and put a HELP message on
it.  */
   /*  Var (char *err_message) is help message.
     */
   /*  This function is calling Draw_Win ().
     */

/*****************************************************************
*****/
   {
   int x_mes, y_mes, xmid;
   x_mes = 100;                                 /* x position of bar
message */
   y_mes = getmaxy() - x_mes;                   /* y position of bar
message */
   xmid  = getmaxx()/2;                        /* middle position for
message */
   /* Draw bar message */
   Draw_Win (x_mes, y_mes, getmaxx() - x_mes, y_mes + 15, MAGENTA);
   settextstyle (2,0,6);
   settextjustify (CENTER_TEXT, TOP_TEXT);     /* Set text to the
middle */
   outtextxy (xmid, y_mes-3, err_message);                /* Write
message */
   settextjustify (LEFT_TEXT, TOP_TEXT);   /* Set text back to the
left */
   }   /* end Bar Message */

void Clr_Mess ()
   /***********************************************************/
   /*  This function is clear message bar and Help message  */
   /*  This function is calling Draw_Win ().                 */
   /*****************************************************************/
   {
   int x_mes, y_mes;
   x_mes = 100;                        /* x position of bar message
*/
   y_mes = getmaxy() - x_mes;          /* y position of bar message
*/
   /* Clear bar message */
   Draw_Win  (x_mes,  y_mes,  getmaxx()  -  x_mes,  y_mes  +  15,
LIGHTBLUE);
   settextjustify (CENTER_TEXT, TOP_TEXT);    /* Set text to the
middle */
   }   /* end Clear Message */

void Cursor (int x, int y, int t_switch)

/********************************************************************/
   /* This function is turn ON/OF the cursor right at x,y position.
*/
   /* var (int x, int y, int t_switch) are x,y position of the
 */
   /*     cursor and the ON/OFF switch.
 */
   /* This function is calling Mouse_Func ()
 */
   /*         and is called by Create_Scrl (),  Create_Scr2 ().
 */

/********************************************************************/
   {
   int curr_color;
   setlinestyle (0,0,2);
   if (t_switch == ON)
      {
      while (!kbhit ())
      {
      setcolor (YELLOW);
      line (x, y, x+8, y);                       /* Draw yellow cursor
*/
      line (x, y+1, x+8, y+1);
      delay (100);                     /* Wait for 1/10 of a second
*/
      setcolor (MAGENTA);
      line (x, y, x+8, y);                   /* Draw background cursor
*/
      line (x, y+1, x+8, y+1);
      delay (100);
      }   /* end while not kbhit */
      }   /* end if */
   else                               /* Switch is OFF turn off the
cursor */
      {
      setcolor (MAGENTA);
      line (x, y, x+8, y);                             /* Draw magenta
cursor */
      line (x, y+1, x+8, y+1);
      }
   setlinestyle (0,0,3);
   }   /* end Cursor */

int norecords ()

/*****************************************************************
****/
   /* This function opens the sfmos.dat file, count records and
closes */
   /* it.  It return the number of records count.
    */

/*****************************************************************
****/
   {
   int stream;
   char ch;
   int rec_cntr = 0;
   if ((stream = open("sfmos.dat", O_CREAT ¦ O_RDWR,
                  S_IREAD ¦ S_IWRITE)) == NULL)  /* Open file */
      {
      closegraph();
      printf("Cannot open sfmos.dat file.\n");
      getch ();
      clrscr ();
      exit (0);
      }
   do
      {
      read (stream, &ch, 1);
      if (ch == '\n')
      rec_cntr++;                /* Count number of record in file
*/
      } while (!eof(stream));       /* end do */
   close (stream);                                       /* Close
file */
   if (rec_cntr <= 0)
      {
      closegraph();
      printf ("error: sfmos.dat file is empty ...");
      getch ();
      exit (0);
      }   /* end if */
   return (rec_cntr);
   }   /* end number of records */

void Warning_Mess ()

/*****************************************************************
**/
   /* This function displays the warning window with warning
```

```
message */
   /* whenever user press F6 to exit or Mouse click on F6 function.
*/
   /* The warning message asks for saving the data entered or not.
 */
   /* This function is called by  Graph_Main_Menu (),
  */
   /*                                Create_Scr1 (),
  */
   /*                                Create_Scr2 ().
  */

/****************************************************************
**/
   {
   void *rect;                /* saves memory of F6 warning window
to RAM */
   int x1, x2, y1, y2;                /* x,y position of F6 warning
window */
   int f6flag;                          /* Set flag to leave while
loop */
   unsigned int size;            /* memory size of warning message
image */
   x1 = getmaxx () - 425;                 /* set x,y exit window
size */
   y1 = getmaxy () - 180;
   x2 = getmaxx () - 25;
   y2 = getmaxy () - 80;
   func_key = FALSE;
   size = imagesize (x1, y1, x2, y2);      /* get memory size of
image */
   rect = malloc (size);
   getimage (x1, y1, x2, y2, rect);
   settextstyle (2,0,6);
   setlinestyle (0,0,2);
   settextjustify (LEFT_TEXT, TOP_TEXT);      /* set text to the
left */
   Draw_Win (x1, y1, x2, y2, LIGHTCYAN);
   setcolor (LIGHTMAGENTA);
   rectangle  (x1+5,y1+5,x2-5,y2-5);                /*  draws  two
rectangles  */
   rectangle (x1+7,y1+7,x2-7,y2-7);           /* inside window
message */
   setcolor (LIGHTCYAN);
   line  (x1+150,y1+5,x2-150,y2-(y2-y1)+5);/*  Draw  two lines with
reverse */
   line  (x1+150,y1+7,x2-150,y2-(y2-y1)+7);/*  color  to  put  'Exit
Program' */
   setcolor (LIGHTMAGENTA);
   outtextxy (x1+159, y1-2, " Warning ");
   outtextxy (x1+160, y1-2, " Warning ");
   outtextxy(x1+20,y1+15,"The total input number of soldiers for");
   outtextxy(x1+20,y1+32,"each course should equal the number of");
   outtextxy(x1+20,y1+49,"soldiers to be assigned.");
   outtextxy(x1+20,y1+70,"Please, Press Enter and Try again.");
   while (((func_key = Get_Fn_Key()) != 13))
      { }   /* end while */
   putimage (x1,y1,rect,COPY_PUT);
   free (rect);
   setlinestyle (0,0,3);
   settextjustify (CENTER_TEXT, TOP_TEXT);
   }   /* end of warning message */

int Check_Key (int screen_no, int no_field,
            int firsthit, char *filename, int modflag)
/*****************************************************************
********/
/*  This function controls function keys such as left, right arrow
key, */
/*  backspace, function key (F1,...,F6).
    */
/*  It is calling Draw_Win(), Cursor(), Bar_Mess(), Help_Win(),
    */
/*  and is called by Entry_Field().
    */
/*****************************************************************
********/
   {
   FILE *field_file;
   int  i;
   int  saveflag = FALSE, renmflag = FALSE;
   switch (func_key)
      {
      case 13:                 /* Enter is pressed for next field
*/

Draw_Win(x_field[curr_pos],y_field[curr_pos],x_field[curr_pos]\
           +len_field,y_field[curr_pos]+18,LIGHTGRAY);
      moveto(x_field[curr_pos]+20,y_field[curr_pos]-3);
      outtext (field_str[curr_pos]);
      curr_pos += 1;
      if (curr_pos > no_field)
         curr_pos = 1;
      if (curr_pos < 1)
         curr_pos = no_field;
      firstkey = TRUE;
      let_cntr = 0;
      Draw_Win (x_field[curr_pos],y_field[curr_pos],\

x_field[curr_pos]+len_field,y_field[curr_pos]+18,MAGENTA);
      moveto(x_field[curr_pos]+20,y_field[curr_pos]-3);
      outtext(field_str[curr_pos]);
      break;
      case 75:           /* Back arrow is pressed for back 1 space
*/
      if (let_cntr >= 1)
         {
         Cursor (xcursor, y_field[curr_pos]+15, OFF);
```

```
          let_cntr--;
          xcursor -= 10;
          Cursor (xcursor, y_field[curr_pos]+15, ON);
          }
       else
          Cursor (xcursor, y_field[curr_pos]+15, ON);
       break;
       case 77:                             /* Forward arrow is pressed
*/
       if (let_cntr < 2)
          {
          Cursor (xcursor, y_field[curr_pos]+15, OFF);
          xcursor += 10;
          Cursor (xcursor, y_field[curr_pos]+15, ON);
          let_cntr++;
          }
       else
          Cursor (xcursor, y_field[curr_pos]+15, ON);
       break;
       case 59:                   /* F1 is pressed for cont function
*/
       if (screen_no == 2)
          {
          func_key = 0;
          Bar_Mess ("There is no next screen");
          messflag = TRUE;
          Cursor (xcursor, y_field[curr_pos]+15, ON);
          }
       else
          {
          if (firsthit == TRUE)
             saveflag = TRUE;
          }
       break;
       case 60:                   /* F2 is pressed for help function
*/
       Help_Win (screen_no);  /* Draw Help Message for screen 1 */
       Cursor (xcursor,y_field[curr_pos]+15,ON);   /* Set cursor on
*/
       func_key = 0;
       break;
       case 61:                   /* F3 is pressed for save function
*/
       func_key = 0;
       Bar_Mess ("Save fields data ...");
       messflag = TRUE;
       saveflag = TRUE;
       Cursor (xcursor,y_field[curr_pos]+15,ON);
       break;
       case 62:                         /* F4 is pressed for prev_scr
function */
       if (screen_no == 1)
          {
          func_key = 0;
          Bar_Mess ("There is no previous screen");
          messflag = TRUE;
          Cursor (xcursor,y_field[curr_pos]+15,ON);
          }
       else
          {
          if (firsthit == TRUE)
             saveflag = TRUE;
          }
       break;
       case 63:                    /* F5 is pressed for Mainmenu
function */
       if (firsthit == TRUE)
          {
          saveflag = TRUE;
          if (modflag == FALSE)
             renmflag = TRUE;
          }
       break;
       case 64:                /* F6 is pressed for exit function
*/
       F6message();
       if (func_key == TRUE)            /* F6message return YES */
          {
          if (firsthit == TRUE)
             {
             if (modflag == FALSE)
             renmflag = TRUE;
             saveflag = TRUE;
             }
          func_key = 64;
          }
       else if (func_key == 2)           /* F6message return NO */
          {
          if ((firsthit == TRUE) && (modflag == FALSE))
             renmflag = TRUE;
          func_key = 64;
          }
       else if (func_key == FALSE)      /* F6message return CANCEL
*/
          Cursor (xcursor,y_field[curr_pos]+15,ON);
       break;
       default: Cursor (xcursor,y_field[curr_pos]+15,ON); break;
       }   /* end switch */
    if (saveflag == TRUE)
       {
       if ((field_file = fopen(filename, "w")) == NULL)
       {
       closegraph();
       printf ("Cannot open %s file.\n", filename);
       getch ();
       exit (0);
       }   /*   end if   */
       for (i = 0; i <= no_field; i++)
```

```
      fprintf (field_file,"%s   \n",field_str[i]);
      fclose (field_file);
      )   /* end if screen_no is 1 */
   if (renmflag == TRUE)
      {
      if (searchpath ("createsl.dat") != NULL)
      {
      if (searchpath ("modifysl.dat") != NULL)
         remove ("modifysl.dat");
      if (rename("createsl.dat", "modifysl.dat") != 0)
         {
         perror("rename");
         getch ();
         closegraph();
         exit(0);
         }   /* end if rename */
      )   /* end if modifile */
      if (searchpath ("creates2.dat") != NULL)
      {
      if (searchpath ("modifys2.dat") != NULL)
         remove ("modifys2.dat");
      if (rename("creates2.dat", "modifys2.dat") != 0)
         {
         perror("rename");
         getch ();
         closegraph();
         exit(0);
         }   /* end if rename */
      )   /* end if modifile */
      )   /* end if renmflag */
   return (func_key);
   }   /* end check key */

int Entry_Field(int no_field, int screen_no, char *filename, int
modflag)
/****************************************************************
*******/
/*  This procedure keep track of function keys, keyboards pressed
and  */
/*  data of the fields on the first and second create parameter
screen */
/*  It is calling       Draw_Win (),             Get_Fn_Key (),
    */
/*                      Check_Key (),            Clr_Mess ()
    */
/*  and is called by    Create_Scrl ().
     */
/****************************************************************
*******/
   {
   FILE *field_file;
   int temp, i, j;
   int temptotl;
   int upd_flag = FALSE;
   int firsthit = FALSE;
   char *spath;
   curr_pos = 1;                                          /* initialize
variables */
   func_key = 0;
   firstkey = TRUE;
   messflag = FALSE;
   let_cntr = 0;
   settextstyle (2,0,6);
   Draw_Win (x_field[curr_pos],y_field[curr_pos],\

x_field[curr_pos]+len_field,y_field[curr_pos]+18,MAGENTA);
   for (i = 0; i <= no_field; i++)        /* initialize array */
      {
      for (j = 0; j < 3; j++)
      field_str[i][j] = ' ';
      field_str[i][3] = '\0';
      }   /* end for i */
   spath = searchpath(filename);
   if (spath != NULL)
      {
      if ((field_file = fopen(filename, "r")) == NULL)
      {
      closegraph();
      printf ("error: can not open data file %s. \n", filename);
      getch ();
      exit (0);
      }   /* end if */
      upd_flag = TRUE;
      settextjustify (CENTER_TEXT, TOP_TEXT);
      for (i = 0; i <= no_field; i++)
      {
      fgets (rec_string, 80, field_file);
      strncpy (field_str[i], rec_string, 3);
      field_str[i][3] = '\0';
      moveto (x_field[i]+20, y_field[i]-3);
      outtext (field_str [i]);
      }
      fclose (field_file);
      }   /* end if spath not null */
   else
      {
      if (screen_no == 1)
      {
      for (i = 5; i <= no_field; i++)
         {
         strncpy (field_str[i], " 80", 3);
         moveto (x_field[i]+20, y_field[i]-3);
         outtext (field_str[i]);
         }
      }   /* end if screen number equal 1 */
      else
      {
      strncpy (field_str[1], "E-5", 3);
```

```
        moveto (x_field[1]+20, y_field[1]-3);
        outtext (field_str[1]);
        for (i = 2; i <= 4; i++)
           {
           strncpy (field_str[i], "E-4", 3);
           moveto (x_field[i]+20, y_field[i]-3);
           outtext (field_str[i]);
           }
        }   /* if screen number not equal 1 */
        }   /* end else of if spath not null */
     while ((func_key != 59) && (func_key != 62) &&
         (func_key  !=  63)  &&  (func_key  !=  64))  /*  func  keys  <>
F(1,4,5,6) */
        {
        settextjustify (CENTER_TEXT, TOP_TEXT);
        if (curr_pos >= no_field)
        upd_flag = TRUE;
        if ((screen_no == 1) && (curr_pos > 4) &&
         (temptotl != total_recs))
        {
        if (upd_flag == FALSE)
                                                D  r  a  w  _  W  i  n
(x_field[curr_pos],y_field[curr_pos],x_field[curr_pos]\
                +len_field,y_field[curr_pos]+18,LIGHTGRAY);
        else
           {
                                                D  r  a  w  _  W  i  n
(x_field[curr_pos],y_field[curr_pos],x_field[curr_pos]\
                +len_field,y_field[curr_pos]+18,LIGHTGRAY);
           moveto(x_field[curr_pos]+20,y_field[curr_pos]-3);
           outtext(field_str[curr_pos]);
           }
        Warning_Mess ();
        curr_pos = 1;
        firstkey = TRUE;
        let_cntr = 0;
        Draw_Win (x_field[curr_pos],y_field[curr_pos],\

x_field[curr_pos]+len_field,y_field[curr_pos]+18,MAGENTA);
        moveto(x_field[curr_pos]+20,y_field[curr_pos]-3);
        outtext(field_str[curr_pos]);
        }
        if (firstkey == TRUE)                       /* first key is
pressed */
        {
        xcursor = x_field[curr_pos]+5;
        Cursor (xcursor,y_field[curr_pos]+15,ON);
        }
        func_key = Get_Fn_Key();
        if ((upd_flag == TRUE) && (firstkey == TRUE))
        {
        Draw_Win (x_field[curr_pos],y_field[curr_pos],\

x_field[curr_pos]+len_field,y_field[curr_pos]+18,LIGHTGRAY);
```

```
       Draw_Win (x_field[curr_pos],y_field[curr_pos],\
x_field[curr_pos]+len_field,y_field[curr_pos]+18,MAGENTA);
       }
       if (firstkey == TRUE)
       firstkey = FALSE;
       if (messflag == TRUE)                        /* set message
flag off */
       {
       Clr_Mess ();
       messflag = FALSE;
       }
       temptotl = 0;
       for (i = 1; i <= 4; i++)
       temptotl += atoi (field_str [i]);
       if  ((is_funckey)  ||  (func_key  ==  13))  /*  function  keys
entered only */
                                f   u   n   c   _   k   e   y          =
Check_Key(screen_no,no_field,firsthit,filename,modflag);
       else  /* keyboard and backspace only.  No function keys */
       {
       Cursor (xcursor,y_field[curr_pos]+15,OFF);
       firsthit = TRUE;
       if (func_key == 8)
          /*  BackSpace  key  is  pressed.   Turn  off  the  cursor  and
erase */
          /* that char then turn on cursor at that new position.
  */
          {
             if ((let_cntr >= 1) && (let_cntr < 3))
             /* if current position is not at the beginning of field
*/
             xcursor -= 10;
             if (let_cntr >= 1)
             {
             let_cntr--;
             field_str[curr_pos][let_cntr] = '\0';
             }
             Draw_Win(xcursor,y_field[curr_pos],\
                xcursor+8,y_field[curr_pos]+14,BLACK);
             Draw_Win(xcursor,y_field[curr_pos],\
                xcursor+8,y_field[curr_pos]+14,MAGENTA);
          }
       else if ((func_key == 9) || (func_key == 27)) {} /* Tab & Esc
*/
       /* checking for integer fields */
       else if ((screen_no == 1) && (curr_pos >= 1) && (curr_pos <=
4) &&
             ((func_key < 48) || (func_key > 57)))
          {
          Bar_Mess ("This is an integer field.");
          messflag = TRUE;
          }
       else if ((screen_no == 2) && (curr_pos >= 5) && (curr_pos <=
8) &&
            ((func_key < 48) || (func_key > 57)))
          {
          Bar_Mess ("This is an integer field.");
          messflag = TRUE;
          }
       else
          {     /* keyboard entered only */
          if (let_cntr < 3)
             {
             Draw_Win(xcursor,y_field[curr_pos],\
                xcursor+8,y_field[curr_pos]+12,BLACK);
             Draw_Win(xcursor,y_field[curr_pos],\
                xcursor+8,y_field[curr_pos]+12,MAGENTA);
              sprintf   (&field_str[curr_pos][let_cntr],    "%c",
func_key);
             outtextxy (xcursor+5,y_field[curr_pos]-3,\
                &field_str[curr_pos][let_cntr]);
             if (let_cntr < 2)
             xcursor += 10;
             let_cntr++;
             }  /* end if let_cntr less than 3 */
          else
             field_str[curr_pos][let_cntr] = '\0';
          }  /* end else */
       Cursor (xcursor,y_field[curr_pos]+15,ON);
       func_key = 0; /* avoid ascii code duplicate keyboard and
func. key */
       }  /* else keyboard and back space and let_cntr >= 1 */
       }  /* end while function key <> F1, F5 and F6 */
    return (func_key);
    }  /* end Entry field */

int Create_Scrl (char *title, char *filename, int modflag)
/*****************************************************************
**********/
/*  This function display the first Data Entry for the create data
file   */
/*  option of the main menu.  It control all function keys from F1
to F6  */
/*  Up, Down, and Enter Key.
       */
/*   This  function  is  calling:           Functn_Bar(),
Entry_Field()     */
/*          and is called by        Graph_Main_Menu().
       */
/*****************************************************************
**********/
    {
    int i,j,k,l,m,h;        /* temp variables */
    int y,x;               /* x,y position for screen line item */
    int distlett;            /* distance between word in function
bars */
    int no_field = 11;      /* number of fields in screen one */
```

```
   int screen_no = 1;       /* screen number */
   char *lineitem [12];     /* Entry screen line item and messages
*/
   char soldier_no[3];

   for (i=1; i<=12; i++)    /* initializes arrays */
      {
      x_field [i] = 0;
      y_field [i] = 0;
      }  /* end for */

   /* Data Entry Messages */
   lineitem [1]  = "    1. Number of soldiers
  ";
   lineitem [2]  = "          Number of soldiers to be assigned =
 ";
   lineitem [3]  = "   2. Input number of soldiers for each course
";
   lineitem [4]  = "    Special Operation Weapon Sergeant (18B) =
 ";
   lineitem [5]  = "  Special Operation Engineer Sergeant (18C) =
 ";
   lineitem [6]  = "   Special Operation Medical Sergeant (18D) =
 ";
   lineitem [7]  = "   Special Operation Commun. Sergeant (18E) =
 ";
   lineitem [8]  = "    3. Input Cut-Scores for each course
  ";
   lineitem [9]  = "   (18B) CO =                (18C) FA =
 CO =";
   lineitem [10] = "   (18D) ST =      GT =      (18E) SC =
 EL =";
   Drawborder ();
   x = (vp.right - vp.left)/2;
   setcolor (YELLOW);
   settextstyle (2,0,6);
   y = 20;
   settextjustify (CENTER_TEXT, TOP_TEXT);
   outtextxy (x+220, y,"(Screen 1 of 2)");          /* Draw screen
number */
   setcolor (LIGHTRED);
   settextstyle (1,0,4);
   h = textheight ("H");
   y = h;
   outtextxy (x,y,title);                           /* Draw screen
title */
   setcolor (YELLOW);
   settextstyle (1,0,3);
   x -= 40;
   y += 1.5*h;
   for (i=1; i<=10; i++)
      {
      outtextxy (x,y,lineitem [i]);          /* output lineitem to
screen */
```

```
        if (i == 2)
        outtextxy (x+235,y,itoa(total_recs,soldier_no,10));
        settextstyle (2,0,6);                    /* set letter style to
normal */
        if ((i == 2) || (i == 7))
        settextstyle (1,0,3);                   /* set letter style for
title */
        if  ((i==1)  ||  (i==3)  ||  (i==8))  /*  set  space  between
title&lineitem */
        y += h;
        else
        y += 0.9*h;
        if ((i >=  ) && (i <= 6))              /* 4 lightbars for 2nd
paragraph */
        {
        Draw_Win (x+220,y,x+220+len_field,y+18,LIGHTGRAY);
        x_field [i-2] = x+220;                 /* get x,y position of
lightbars */
        y_field [i-2] = y;
        }
        k = 120;
        l = 155;
        if (i == 8)                           /* set position for the 8
lightbars */
        {
        Draw_Win (x-k,y,x-k+len_field,y+18,LIGHTGRAY);
        x_field [5] = x-k;                     /* get x,y positions of
lightbars */
        y_field [5] = y;
        k -= 110;
        Draw_Win (x+l,y,x+l+len_field,y+18,LIGHTGRAY);
        x_field [6] = x+l;     /* get x,y positions of lightbars */
        y_field [6] = y;
        l += 110;
        Draw_Win (x+l,y,x+l+len_field,y+18,LIGHTGRAY);
        x_field [7] = x+l;     /* get x,y positions of lightbars */
        y_field [7] = y;
        l += 110;
        }
        if (i == 9)
        {
        m = 7;
        for (j=1; j<=2; j++)          /* draw 8 lightbars for 3rd
paragraph */
            {
            Draw_Win (x-k,y,x-k+len_field,y+18,LIGHTGRAY);
            x_field [j+m] = x-k;              /* get x,y positions of
lightbars */
            y_field [j+m] = y;
            k -= 110;
            Draw_Win (x+l,y,x+l+len_field,y+18,LIGHTGRAY);
            x_field [j+m+2] = x+l;           /* get x,y positions of
lightbars */
            y_field [j+m+2] = y;
```

```
          l += 110;
          }   /* end for j = 1 */
       }   /* end if */
       }   /*   end for i = 1  */
   distlett = 50;                /* set distance between word on
function bars */
   Functn_Bar (distlett);                                /* called
function bars */
   return (Entry_Field(no_field, screen_no, filename, modflag));
   }   /* end create_scr1 */

int Create_Scr2 (char *title, char *filename, int modflag)
/*******************************************************************
**********/
/*  This function displays the Second Data Entry for the create
data file */
/*  option of the main menu.  It controls all function keys from F1
to F6 */
/*  Up, Down, and Enter Key.
       */
/*   This  function  is  calling:            Functn_Bar(),
Entry_Field()     */
/*           and is called by         Create_Data_File().
       */
/*******************************************************************
**********/
   {
   int i,j,k,l,m,h,y,x;     /* temporary variables */
   int distlett;             /* distance between word in function
bars */
   int no_field = 8;        /* number of field in screen two */
   int screen_no = 2;       /* screen number */
   char *lineitm2 [10];     /* Entry screen line item and messages
*/

   messflag = 0;
   for (i=1; i<=10; i++)    /* initializes arrays */
      {
      x_field [i] = 0;
      y_field [i] = 0;
      }   /*   end for   */
   lineitm2 [1]  = "    4. Input grade restriction for each course
 ";
   lineitm2 [2]  = "              (18B)
  ";
   lineitm2 [3]  = "              (18C)
  ";
   lineitm2 [4]  = "              (18D)
  ";
   lineitm2 [5]  = "              (18E)
  ";
   lineitm2 [6]   = "         5.  Input  MOS  preference  (1=Low  ->
10=High) ";
   lineitm2 [7]  = "              (18B)            (18C)
```

```
";
 lineitm2 [8]  = "                 (18D)              (18E)
";
 Drawborder ();
 x = (vp.right - vp.left)/2;
 setcolor (YELLOW);
 settextstyle (2,0,6);
 settextjustify (CENTER_TEXT, TOP_TEXT);
 y = 20;
 outtextxy (x+220, y,"(Screen 2 of 2)");
 setcolor (LIGHTRED);
 settextstyle (1,0,4);
 h = textheight ("H");
 y = h;
 outtextxy (x,y,title);
 setcolor (YELLOW);
 settextstyle (1,0,3);
 x -= 40;
 y += 2*h;
 for (i=1; i<=8; i++)
    {
    outtextxy (x,y,lineitm2 [i]);
    settextstyle (2,0,6);
    if ((i == 5) || (i == 8))
    settextstyle (1,0,3);
    if ((i == 1) || (i == 6))
    y += 1.5*h;
    else
    y += h;
    if (i <= 4)                /* 4 lightbars for 4th paragraph */
    {
    Draw_Win (x-25,y,x-25+len_field,y+18,LIGHTGRAY);
    x_field [i] = x-25;
    y_field [i] = y;
    }  /* end if i <= 4 */
    if ((i >= 6) && (i <= 7))  /* 4 lightbars for 5th paragraph
*/
    {
    if (i == 6)
       m = 4;
    else
       m = 6;
    k = -50;
    for (j=1; j<=2; j++)
       {
       Draw_Win (x+k,y,x+k+len_field,y+18,LIGHTGRAY);
       x_field [j+m] = x+k;
       y_field [j+m] = y;
       k += 170;
       }  /* end for */
    }  /* end if i >= 6 */
    }  /*  end for i = 1  */
 distlett = 50;
 Functn_Bar (distlett);
```

```
   return (Entry_Field(no_field, screen_no, filename, modflag));
   }   /* end create_scr2 */

int Create_Data_File (void)
/*****************************************************************
**********/
/*  This function calls two Data Entry screen for the create data
file    */
/*  option of the main menu.
       */
/*   This function is calling:            Create_Scr1(),
Create_scr2()    */
/*          and is called by        Graph_Main_Menu().
       */
/*****************************************************************
**********/
   {
   char *title;
   char *filename;
   int modflag = FALSE;
   title = "CREATE PARAMETER DATA FILE";
   func_key = 62;
   while ((func_key != 63) && (func_key != 64))   /* func_key <>
F5, F6 */
      {
      switch (func_key)
      {
      case 59:
         filename = "creates2.dat";
         func_key = Create_Scr2 (title, filename, modflag); break;
      case 62:
         filename = "creates1.dat";
         func_key = Create_Scr1 (title, filename, modflag); break;
      }   /* end switch */
      }   /* end while */
   return (func_key);
   }   /* end create data file */



int Modify_Data_File (void)
/*****************************************************************
**********/
/*  This function calls two Data Entry screen for the create data
file    */
/*  option of the main menu.
       */
/*   This function is calling:            Create_Scr1(),
Create_scr2()    */
/*          and is called by        Graph_Main_Menu().
       */
/*****************************************************************
**********/
```

```
   {
   char *title;
   char *filename;
   int modflag = TRUE;
   title = "MODIFY PARAMETER DATA FILE";
   func_key = 62;
   while ((func_key != 63) && (func_key != 64))   /* func_key <>
F5, F6 */
      {
      switch (func_key)
      {
      case 59:
         filename = "modifys2.dat";
         func_key = Create_Scr2 (title, filename, modflag); break;
      case 62:
         filename = "modifys1.dat";
         func_key = Create_Scr1 (title, filename, modflag); break;
      }   /* end switch */
      }   /* end while */
   return (func_key);
   }   /* end modify_data_file */

void Get_Data()
/*****************************************************************
*********/
/* This function reads data from file C_ARRAY.DAT and puts them
into an  */
/* array of strings, and then converts string data into floating
points. */
/*  All  converted  floating  points  are  inserted  into  an  two
dimensional     */
/*  global  array  for  later  uses.   A  global  array  contains
information of  */
/* c[], constant[], coefaa1[], coefaa2[], coefmo[], coefw[].
      */
/* This function is called by Run_Model().
      */
/*****************************************************************
*********/
   {
   FILE *c_file;
   char buffer[11];
   int cnumb, cntr, temp;

   if ((c_file = fopen("c_array.dat", "r")) == NULL)
      {
      closegraph();
      printf ("error: can not open data file c_array.dat.\n");
      getch ();
      exit(0);
      }   /*   end if   */
   cntr = 1;
   for (temp = 1; temp <= 6; temp++)        /* converses string to
float */
```

```
        {
        fgets(rec_string, 80, c_file);
        for (cnumb = 1; cnumb <= 40; cnumb++)
        {
        buffer[cntr] = rec_string[cnumb];
        cntr++;
        if ((cnumb % 10) == 0)
           {
           buffer[0] = 10;
           buffer[cntr] = '\0';
           ntable[temp][cnumb/10] = atof(buffer);
           cntr = 1;
           }  /*   end if   */
        }  /*   for cnumb = 1   */
        }  /*   for temp = 1   */
    fclose(c_file);
    }  /*   end get data procedure   */

float Get_Field (int data_position)
/*****************************************************************
*********/
/*  This function gets a position of data from a record string,
converts */
/*  it from characters into float and then return it.
     */
/*  It is called by  Average_For(),      Standard_For(),
     */
/*                   E_Power(),          Assign_18().
     */
/*****************************************************************
*********/
    {
    char buffer[3];
    buffer[0] = rec_string[data_position];
    buffer[1] = rec_string[data_position+1];
    buffer[2] = rec_string[data_position+2];
    buffer[3] = '\0';
    return(atof(buffer));
    }  /*   end get field   */

float Average_For (const data_position)
/*****************************************************************
*********/
/*  This function gets data from SFMOS.DAT file and then converts
data   */
/*   from the specify position into integer and calculate the
average of  */
/*  them.  The data_position is for score of CO, FA, ST, or EL.
     */
/*  It is calling     Get_Field()
     */
/*  and is called by  Standar_For(),        Assign_18().
     */
/*****************************************************************
```

```
*********/
   {
   FILE *s_file;
   float total_score=0.0;
   int cntr;
   if ((s_file = fopen("sfmos.dat", "r")) == NULL)
      {
      closegraph();
      printf ("error: can not open data file sfmos.dat.\n");
      getch ();
      exit(0);
      }  /*   end if   */

   for (cntr = 1; cntr <= total_recs; cntr++)
      {                                       /* calculate total of EL */
      fgets(rec_string, 80, s_file);
      total_score = total_score + Get_Field(data_position);
      }
   fclose(s_file);
   return (total_score/(float)total_recs);
   }  /*   end average for   */

float Standard_For(const data_position)
/*******************************************************************
*******/
/*   This function calculates the standard score by using the
equation  */
/*  StandardScore = square root of(Score(i) - AverageScore)square
    */
/*  divided by square root of(N(records) minus 1).
    */
/*  Standard = sqrt((X(i)-Avg(X))*(X(i)-Avg(X))) / sqrt(N(rec)-1)
    */
/*  It is calling    Get_Field()      Average_For()
    */
/*  and is called by   Assign_18().
    */
/*******************************************************************
*******/
   {
   FILE *s_file;
   float total_score=0.0;
   float average_score;
   int cntr;
   average_score = Average_For(data_position);
   if ((s_file = fopen("sfmos.dat", "r")) == NULL)
      {
      closegraph();
      printf ("error: can not open data file sfmos.dat.\n");
      getch ();
      exit(0);
      }  /*   end if   */
   for (cntr = 1; cntr <= total_recs; cntr++)
      {
```

```
      fgets(rec_string, 80, s_file);
      total_score = total_score +
              sqrt((Get_Field(data_position)-average_score)*
                (Get_Field(data_position)-average_score));
      }
   fclose(s_file);
   return (total_score/(sqrt((float)total_recs-1)));
   }  /*   end Standard_For    */

float E_Power (int array_pos, float score_xx)
/*******************************************************************
******/
/*  This function calculates the value of 'e' to the power of <exp>
 */
/*  to estimate the standard deviation for all elements of the set
  */
/*  of ASVAB = (CO, FA, GT, ST, EL, SC) scores.
   */
/*  It is calling    Get_Field(),
   */
/*  and is called by Assign_18x().
   */
/*******************************************************************
******/
   {
   float f_score, com_hea, wants_n, t_power;
   char t_string[2];
   t_string[0] = rec_string[44];
   t_string[1] = rec_string[45];
   t_string[2] = '\0';
   switch (array_pos)
      {
      case 1:
      f_score = 0.0;
      if ((atoi(t_string) >= 11) || (atoi(t_string) <= 19))
         com_hea = 1.0;
      else com_hea = 0.0;
      if (rec_string[43] == 'B') wants_n = 1.0; else wants_n = 0.0;
      break;
      case 2:
      f_score = Get_Field(B_CO);
      if ((atoi(t_string) >= 11) || (atoi(t_string) <= 19))
         com_hea = 1.0;
      else com_hea = 0.0;
      if (rec_string[43] == 'C') wants_n = 1.0; else wants_n = 0.0;
      break;
      case 3:
      f_score = Get_Field(D_GT);
      if ((atoi(t_string) == 91) || (atoi(t_string) == 92))
         com_hea = 1.0;
      else com_hea = 0.0,
      if (rec_string[43] == 'D') wants_n = 1.0; else wants_n = 0.0;
      break;
      case 4:
        f_score = Get_Field(E_SC);
        com_hea = 0.0;
        if (rec_string[43] == 'E') wants_n = 1.0; else wants_n = 0.0;
        break;
        }   /*   end switch   */

    t_power = (-1.0)*(ntable[2][array_pos]+
                    ntable[3][array_pos]*score_xx+
                    ntable[4][array_pos]*f_score+
                    ntable[5][array_pos]*com_hea+
                    ntable[6][array_pos]*wants_n);

    return(exp(t_power));
    }   /*   end e power   */

void Get_Weight ()
/*****************************************************************
********/
/* This function reads data from file WEIGHT.DAT and puts them into
an  */
/* array of strings and then converts string data into floating
points. */
/*  All  converted  floating  points  are  inserted  into  an  two
dimensional   */
/*  global  array  for  later  uses.   A  global  array  contains
information    */
/* for the weight of the objective of the model.
    */
/* This function is called by Assign_18x().
    */
/*****************************************************************
********/
    {
    FILE *mod2_file;
    int temp;

    if ((mod2_file = fopen("modifys2.dat", "r")) == NULL)
        {
        closegraph();
        printf ("error: can not open data file modifys2.dat.\n");
        getch ();
        exit(0);
        }   /*   end if   */
    for (temp = 1; temp <= 4; temp++)
        {
        wtable[1][temp] = 0.0;
        }
    for (temp = 1; temp <= 5; temp++)
        fgets(rec_string, 80, mod2_file);

    for (temp = 1; temp <= 4; temp++)
        {
        fgets(rec_string, 40, mod2_file);
        wtable[2][temp] = (-1)*atof(rec_string);
```

```
      }  /*   for temp = 1   */
   fclose(mod2_file);
   }  /*   end get weight procedure   */

void Assign_18x(int choice, int recs_used)
/*****************************************************************
****/
/*  This function generates the column of LINDO input file
 */
/*  It is calling      Average_For(),    Standard_For(),
 */
/*                     Get_Field(),      E_Power()
 */
/*  and is called by   Gen_MPS_File().
 */
/*****************************************************************
****/
   {
   FILE *s_file;
   FILE *inp_file;
   int  cntr1, cntr2, i, j;
   char *f_spaces;
   char *b_spaces = "           ";
   float score_CO,   score_FA,   score_ST,   score_EL;
   float average_B,  average_C,  average_D,  average_E;
   float standard_B, standard_C, standard_D, standard_E;

   average_B  = Average_For(B_CO);
   average_C  = Average_For(C_FA);
   average_D  = Average_For(D_ST);
   average_E  = Average_For(E_EL);

   standard_B = Standard_For(B_CO);
   standard_C = Standard_For(C_FA);
   standard_D = Standard_For(D_ST);
   standard_E = Standard_For(E_EL);

   Get_Weight ();
   if ((s_file = fopen("sfmos.dat", "r")) == NULL)
      {
      closegraph();
      printf ("error: can not open data file sfmos.dat.\n");
      getch ();
      exit(0);
      }  /*   end if   */
   if ((inp_file = fopen("inp_file.dat", "a")) == NULL)
      {
      closegraph();
      printf ("Cannot open output file.\n");
      getch ();
      exit (0);
      }  /*   end if   */
   cntr1 = 1;
   for (i = 1; i <= total_recs; i++)
```

```
{
if (i < 10)
f_spaces = "       ";
else
if (i < 100)
f_spaces = "      ";
else
f_spaces = "     ";

fgets(rec_string, 80, s_file);
score_CO = Get_Field(B_CO);
score_FA = Get_Field(C_FA);
score_ST = Get_Field(D_ST);
score_EL = Get_Field(E_EL);
cntr1++;
cntr2 = 2;
for (j = 1; j <= 4; j++)
{
cntr2++;
if ((check_table[j][i] == 0) && (check_table[j+4][i] == 0))
   {
   switch (choice)
      {
      case 1:
      switch (j)
         {
         case 1:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
         i,j,f_spaces,b_spaces,ntable[1][1]*score_CO);
         break;
         case 2:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
         i,j,f_spaces,b_spaces,ntable[1][2]*score_FA);
         break;
         case 3:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
         i,j,f_spaces,b_spaces,ntable[1][3]*score_ST);
         break;
         case 4:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
         i,j,f_spaces,b_spaces,ntable[1][4]*score_EL);
         break;
         }
      break;
      case 2:
      switch (j)
         {
         case 1:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
         i,j,f_spaces,b_spaces,\
         ntable[1][1]*((score_CO-average_B)/standard_B));
         break;
         case 2:
         fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   ntable[1][2]*((score_FA-average_C)/standard_C));
   break;
   case 3:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   ntable[1][3]*((score_ST-average_D)/standard_D));
   break;
   case 4:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   ntable[1][4]*((score_EL-average_E)/standard_E));
   break;
   }
break;
case 3:
switch (j)
   {
   case 1:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   (1.0/(1.0+E_Power(1, score_CO)))); break;
   case 2:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   (1.0/(1.0+E_Power(2, score_FA)))); break;
   case 3:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   (1.0/(1.0+E_Power(3, score_ST)))); break;
   case 4:
   fprintf (inp_file,"    X%d%d%s1%s%12.7f\n",\
   i,j,f_spaces,b_spaces,\
   (1.0/(1.0+E_Power(4, score_EL)))); break;
   }
break;
}  /*  end switch   */
if (cntrl < 10)
{
fprintf (inp_file,"    X%d%d%s%d%s   1.0000000\n",\
      i,j,f_spaces,cntrl,b_spaces);
if (recs_used+2 <100)
   b_spaces = "        ";
else b_spaces = "       ";
fprintf (inp_file,"    X%d%d%s%d%s   1.0000000\n",\
      i,j,f_spaces,recs_used+2,b_spaces);
fprintf (inp_file,"    X%d%d%s%d%s   1.0000000\n",\
      i,j,f_spaces,recs_used+cntr2,b_spaces);
}
else if (cntrl < 100)
{
b_spaces = "        ";
fprintf (inp_file,"    X%d%d%s%2d%s   1.0000000\n",\
      i,j,f_spaces,cntrl,b_spaces);
```

```
          if (recs_used+2 <100)
             b_spaces = "         ";
          else b_spaces = "        ";
          fprintf (inp_file,"    X%d%d%s%2d%s   1.0000000\n",\
                i,j,f_spaces,recs_used+2,b_spaces);
          fprintf (inp_file,"    X%d%d%s%2d%s   1.0000000\n",\
                i,j,f_spaces,recs_used+cntr2,b_spaces);
          }
          else if (cntr1 < 1000)
          {
          b_spaces = "        ";
          fprintf (inp_file,"    X%d%d%s%3d%s   1.0000000\n",\
                i,j,f_spaces,cntr1,b_spaces);
          fprintf (inp_file,"    X%d%d%s%3d%s   1.0000000\n",\
                i,j,f_spaces,recs_used+2,b_spaces);
          fprintf (inp_file,"    X%d%d%s%3d%s   1.0000000\n",\
                i,j,f_spaces,recs_used+cntr2,b_spaces);
          }   /* end if */
       b_spaces = "          ";
       }   /* end if check_table */
    }   /*   end for j   */
    }   /*   end for i   */
 i = 2;
 for (j = 1; j <= 4; j++)
    {
    i++;
    if (recs_used < 100)
    {
    fprintf (inp_file,"    DP%d%s 1%s%12.7f\n",\
          j,f_spaces,b_spaces,wtable[1][j]);
    fprintf (inp_file,"    DP%d%s %2d%s -1.0000000\n",\
          j,f_spaces,recs_used+i,b_spaces);
    }
    else if (recs_used < 1000)
    {
    fprintf (inp_file,"    DP%d%s  1%s%12.7f\n",\
          j,f_spaces,b_spaces,wtable[1][j]);
    fprintf (inp_file,"    DP%d%s  %3d%s-1.0000000\n",\
          j,f_spaces,recs_used+i,b_spaces);
    }
    }   /* end for i */
 i = 2;
 for (j = 1; j <= 4; j++)
    {
    i++;
    if (recs_used < 100)
    {
    fprintf (inp_file,"    DM%d%s 1%s%12.7f\n",\
          j,f_spaces,b_spaces,wtable[2][j]);
    fprintf (inp_file,"    DM%d%s %2d%s  1.0000000\n",\
          j,f_spaces,recs_used+i,b_spaces);
    }
    else if (recs_used < 1000)
    {
        fprintf (inp_file,"    DM%d%s  1%s%12.7f\n",\
              j,f_spaces,b_spaces,wtable[2][j]);
        fprintf (inp_file,"    DM%d%s  %3d%s 1.0000000\n",\
              j,f_spaces,recs_used+i,b_spaces);
        }
        }   /* end for i */
     fclose(inp_file);
     fclose(s_file);
     }   /*  end of assign 18   */

  void Gen_MPS_File (int choice)
  /*********************************************************************
  ******/
  /*  This function generates MPS file for LINDO input file.
     */
  /*  It is calling   Assign_18x().
     */
  /*********************************************************************
  ******/
     {
     FILE *inp_file;
     int cntr, i, j;
     float sold_arr[5];
     float score_arr[10];
     int   grade_arr[5][5];

     for (i = 1; i <= 4; i++)
        {
        sold_arr[i]  = 0.0;
        for (j = 1; j <= 4; j++)
        grade_arr[i][j] = ' ';
        }
     for (i = 1; i <= 8; i++)
        score_arr[i] = 0.0;
     for (i = 1; i <= 9; i++)       /* initialize array */
        {
        for (j = 1; j <= total_recs; j++)
        check_table [i][j] = 0;
        }   /* end for i */

     if ((inp_file = fopen("modifys1.dat", "r")) == NULL)
        {
        closegraph();
        printf ("Cannot open modifys1.dat file.\n");
        getch ();
        exit (0);
        }   /*   end if   */
     fgets (rec_string, 80, inp_file);
     for (i = 1; i <= 4; i++)   /* Get number of soldiers from
  modifys1 file */
        {
        fgets (rec_string, 80, inp_file);
        sold_arr [i] = atof (rec_string);
        }   /* end for */
```

```
   for (i = 1; i <= 7; i++)  /* Get number of cut-score from
modifys1 file */
      {
      fgets (rec_string, 80, inp_file);
      score_arr [i] = atof (rec_string);
      }  /* end for */
   fclose (inp_file);

   if ((inp_file = fopen("modifys2.dat", "r")) == NULL)
      {
      closegraph();
      printf ("Cannot open modifys2.dat file.\n");
      getch ();
      exit (0);
      }  /*   end if   */
   fgets (rec_string, 80, inp_file);
   for (i = 1; i <= 4; i++)  /* Get grade E-# from modifys2.dat
file */
      {
      fgets (rec_string, 80, inp_file);
      grade_arr [i][1] = rec_string[0];
      grade_arr [i][2] = rec_string[1];
      grade_arr [i][3] = rec_string[2];
      grade_arr [i][4] = '\0';
      }  /* end for */
   fclose (inp_file);

   if ((inp_file = fopen("sfmos.dat", "r")) == NULL)
      {
      closegraph();
      printf ("Cannot open output file.\n");
      getch ();
      exit (0);
      }  /*   end if   */

   for (i = 1; i <= total_recs; i++)
      {
      fgets (rec_string, 80, inp_file);
      if (Get_Field(B_CO) < score_arr[1])
      check_table [1][i] = 1;
      if ((Get_Field(C_FA) < score_arr[2]) ||
       (Get_Field(B_CO) < score_arr[3]))
      check_table [2][i] = 1;
      if ((Get_Field(D_ST) < score_arr[4]) ||
       (Get_Field(D_GT) < score_arr[5]))
      check_table [3][i] = 1;
      if ((Get_Field(E_SC) < score_arr[6]) ||
       (Get_Field(E_EL) < score_arr[7]))
      check_table [4][i] = 1;
      for (j = 1; j <= 4; j++)
      {
      if (rec_string[B_CO - 1] < grade_arr [j][3])
         check_table [j+4][i] = 1;
      }
```

```
     }
  fclose (inp_file);

  if ((inp_file = fopen("inp_file.dat", "w")) == NULL)
     {
     closegraph();
     printf ("Cannot open inp_file.dat file.\n");
     getch ();
     exit (0);
     }  /*   end if   */
  fprintf (inp_file,"NAME     LINDO GENERATED MPS FILE (MAX)\n");

  fprintf (inp_file,"ROWS\n");                    /* generates rows
*/
  fprintf (inp_file,"  N 1\n");

  for (i = 1; i <= total_recs; i++)
     {
     if (((check_table[1][i] == 1) || (check_table[5][i] == 1)) &&
      ((check_table[2][i] == 1) || (check_table[6][i] == 1)) &&
      ((check_table[3][i] == 1) || (check_table[7][i] == 1)) &&
      ((check_table[4][i] == 1) || (check_table[8][i] == 1)))
     check_table[9][i] = 1;
     }  /* end for */
  cntr = 0;

  for (i = 1; i <= total_recs; i++)
     {
     cntr++;
     if (check_table[9][i] == 0)
     fprintf (inp_file,"  E %d\n", cntr+1);
     else
     cntr--;
     }
  for (i = 1; i <= 5; i++)
     fprintf (inp_file,"  E %d\n", cntr+i+1);

  fprintf (inp_file,"COLUMNS\n");
  fclose(inp_file);

  Assign_18x(choice,cntr);          /* generates columns */

  if ((inp_file = fopen("inp_file.dat", "a")) == NULL)
     {
     closegraph();
     printf("Cannot open inp_file.dat file.\n");
     getch ();
     exit (0);
     }  /*   end if   */
  fprintf(inp_file,"RHS\n");  /* generates RHS right hand side */

  for (i = 1; i <= cntr; i++)
     {
     if (i+1 < 10)
```

```
      fprintf(inp_file,"       RHS             %d
1.0000000\n",i+1);
      else if (i+1 < 100)
      fprintf(inp_file,"       RHS             %2d
1.0000000\n",i+1);
      else
      fprintf(inp_file,"       RHS              %3d
1.0000000\n",i+1);
      }
   if (cntr+2 < 100)
      {
      fprintf(inp_file,"    RHS        %2d           %12.7f\n",\
           cntr+2,(float)total_recs);
      for (i = 1; i <= 4; i++)
      fprintf(inp_file,"    RHS        %2d           %12.7f\n",\
           cntr+2+i,sold_arr[i]);
      }
   else if (total_recs+2 < 1000)
      {
      fprintf(inp_file,"    RHS        %3d         %12.7f\n",\
           cntr+2,(float)total_recs);
      for (i = 1; i <= 4; i++)
      fprintf(inp_file,"    RHS        %3d         %12.7f\n",\
           cntr+2+i,sold_arr[i]);
      }

   fprintf(inp_file,"ENDATA\n");
   fclose(inp_file);
   }  /*  end generated MPS file   */

int Run_Model(void)
/*****************************************************************
******/
/*  This function runs the Special Forces MOS Allocation Model to
   */
/*  determine the number of soldiers to be assigned, estimates the
  */
/*  means for all elements of the {CO, FA, GT, ST, EL, SC} = the
set  */
/*  of ASVAB scores and estimate the standard deviation for all
   */
/*  elements of the set of ASVAB scores.
   */
/*  It is calling   Gen_MPS_File()
   */
/*  and is called by Main_Menu_Key().
   /
/*****************************************************************
******/
   {
   int h,y,x;
   int i_choice = 3;
   int distlett = 50;
   Drawborder ();
```

```
settextjustify (CENTER_TEXT, TOP_TEXT);
x = (vp.right - vp.left)/2;
setcolor (LIGHTRED);
settextstyle (1,0,4);
h = textheight ("H");
y = h;
outtextxy (x,y,"RUN MOS-ARI MODEL");
y += 3*h;
x = 320;
setcolor (YELLOW);
settextstyle (2,0,6);
outtextxy (x+2,y,"     ENTER                    ");
outtextxy (x+3,y,"     ENTER                    ");
outtextxy (x,y,"PRESS ENTER TO RUN MOS-ARI MODEL");
y += 2*h;
Functn_Bar (distlett);
func_key = 0;
messflag = FALSE;
while ((func_key != 63) && (func_key != 64))
   {
   settextstyle (2,0,6);
   setcolor (YELLOW);
   func_key = Get_Fn_Key ();
   if (messflag == TRUE)
   {
   Clr_Mess ();
   messflag = FALSE;
   }
   switch (func_key)
   {
   case 13:                  /* enter key is pressed */
      outtextxy (x,y,"Creating inp_file.dat file for LINDO input
...");
      sleep (1);
      Get_Data ();
      Gen_MPS_File (i_choice);
      y += 2*h;
      outtextxy (x,y,"Running LINDO.  Please, wait ...");
      sleep(1);
      if (system ("LINDO > OUT_FILE.DAT") == -1)
         {
         Clr_Mess ();
         Bar_Mess ("Error while running LINDO. Program abort");
         getch ();
         closegraph ();
         exit (0);
         }
      messflag = TRUE;
      func_key = 63;
      break;
   case 59:
      Bar_Mess ("There is no next screen");
      messflag = TRUE;
      break;
      case 60: Help_Win (1); break;      /* F2 is pressed for help
*/
      case 61:                           /* F3 is pressed for save
*/
         Bar_Mess ("Save is not available here");
         messflag = TRUE;
         break;
      case 62:                        /* F4 is pressed for prev */
         Bar_Mess ("There is no previous screen");
         messflag = TRUE;
         break;
      case 63: break;                   /* F5 is pressed for main
menu */
      case 64:
         F6message();
         if (func_key == TRUE)          /* F6message return YES */
            func_key = 64;
         else if (func_key == 2)        /* F6message return NO */
            func_key = 64;
         else if (func_key == FALSE)    /* F6message return CANCEL
*/
            break;
      }   /* end switch */
      }   /* end while */
   return (func_key);
   }   /* end run model */

int Sumary_Data (int data_position)
/*****************************************************************
*********/
/*  This function gets a position of data from a record string,
converts */
/*  it from characters into int and then returns it.
      */
/*  It is called by  Average_For()       Standard_For()
      */
/*                   E_Power()           Assign_18().
      */
/*****************************************************************
*********/
   {
   char buffer[3];
   buffer[0] = rec_string[data_position];
   buffer[1] = rec_string[data_position+1];
   buffer[2] = rec_string[data_position+2];
   buffer[3] = '\0';
   return(atoi(buffer));
   }  /*   end get field   */

int Sumary_Result ()
/*****************************************************************
******/
/*  This function gets the result of the integer program (LINDO)
and  */
```

```
/*  displays them on the screen.
   */
/*  It is calling Func_Bar();
   */
/*  and is called by Sub_Menu_Key().
   */
/*****************************************************************
******/
   {
   FILE *out_file;
   char buffer[arr_len];
   char *findstr = "VARIABLE        VALUE";
   char *tempptr, tempchar = 'X';
   char dumchar[20];
   int  cntrl, cntr2;
   int  i,h,x,y;
   int  distlett = 50;
   char *wantnumb[5];
   int  sum_arr[5][12];
   int  match_arr[max_recs];
   int  tempflag, recsflag;

   for (cntrl = 0; cntrl <= max_recs-10; cntrl++)
      match_arr[cntrl] = 0;

   for (cntrl = 1; cntrl <= 4; cntrl++)
      for (cntr2 = 1; cntr2 <= 10; cntr2++)
      sum_arr[cntrl][cntr2] = 0;

   wantnumb[1] = "18B";
   wantnumb[2] = "18C";
   wantnumb[3] = "18D";
   wantnumb[4] = "18E";
   settextjustify (CENTER_TEXT, TOP_TEXT);
   Drawborder();
   x = (vp.right - vp.left)/2;
   setcolor(LIGHTRED);
   settextstyle(1,0,4);
   h = textheight("H");
   y = 1.5*h;
   outtextxy (x,y,"SUMARY OF THE RESULT");
   setcolor (YELLOW);
   y += 2*h;
   settextstyle (1,0,3);
   outtextxy (x,y,"Course Soldiers Match  CO  EL  FA  GT  SC  ST");
   y += 1.5*h;
   settextstyle (2,0,6);

   if ((out_file = fopen ("out_file.dat", "r+")) == NULL)
      {
      closegraph();
      printf ("cannot open out_file.dat file. \n");
      getch ();
      exit (0);
      }
   fgets (rec_string, 80, out_file);

   while (strstr (rec_string, findstr) == NULL)
      {
      fgets (rec_string, 80, out_file);
      if (feof(out_file))
      {
      Bar_Mess ("Error in OUT_FILE.DAT file.  Run LINDO again.");
      getch ();
      fclose (out_file);
      return (62);
      }
      }   /* end while */

   findstr = "1.000000";
   recsflag = FALSE;
   while (recsflag == FALSE)
      {
      tempflag = FALSE;
      while (tempflag == FALSE)
      {
      fgets (rec_string, 80, out_file);
      if (feof(out_file))
         {
         fclose (out_file);
         closegraph();
         printf  ("Error  in  OUT_FILE.DAT  file.    See  your
Administrator.");
         getch ();
         exit (0);
         }
      tempptr = strchr (rec_string, tempchar);
      if (!tempptr)
         {
         recsflag = TRUE;
         tempflag = TRUE;
         continue;
         }
      if (strstr (rec_string, findstr) != NULL)
         {
         tempptr = strchr (rec_string, tempchar);
         if (tempptr)
            {
            for (cntr2 = 0; cntr2 <= 5; cntr2++)
           buffer[cntr2] = rec_string[tempptr-rec_string+cntr2+1];
            buffer[6] = '\0';
            cntr2 = (atoi(buffer) % 10);
            cntr1 = ((atoi(buffer) - cntr2) / 10) -1;
            }
         else
            {
            fclose (out_file);
            closegraph ();
```

APPENDIX B. INPUT AND OUTPUT FILES FOR THE SFMOS SYSTEM

```
NAME      LINDO GENERATED MPS FILE (MAX)
ROWS
  N 1
  E 2
  E 3
  E 4
  E 5
  E 6
  E 7
  E 8
  E 9
  E 10
  E 11
  E 12
  E 13
  E 14
  E 15
  E 16
  E 17
  E 18
  E 19
  E 20
  E 21
  E 22
  E 23
  E 24
  E 25
  E 26
COLUMNS
    X12        1             0.0000000
    X12        2             1.0000000
    X12        22            1.0000000
    X12        24            1.0000000
    X13        1             0.0111690
    X13        2             1.0000000
    X13        22            1.0000000
    X13        25            1.0000000
    X14        1             0.0016335
    X14        2             1.0000000
    X14        22            1.0000000
    X14        26            1.0000000
    X22        1             0.0000000
    X22        3             1.0000000
    X22        22            1.0000000
    X22        24            1.0000000
    X23        1             0.5738599
    X23        3             1.0000000
    X23        22            1.0000000
    X23        25            1.0000000
    X24        1             0.0149673
    X24        3             1.0000000
    X24        22            1.0000000
    X24        26            1.0000000
    X31        1             1.0000000
```

X31	4	1.0000000
X31	22	1.0000000
X31	23	1.0000000
X32	1	0.0000000
X32	4	1.0000000
X32	22	1.0000000
X32	24	1.0000000
X33	1	0.0444092
X33	4	1.0000000
X33	22	1.0000000
X33	25	1.0000000
X34	1	0.4854517
X34	4	1.0000000
X34	22	1.0000000
X34	26	1.0000000
X41	1	1.0000000
X41	5	1.0000000
X41	22	1.0000000
X41	23	1.0000000
X42	1	0.0000000
X42	5	1.0000000
X42	22	1.0000000
X42	24	1.0000000
X43	1	0.0617233
X43	5	1.0000000
X43	22	1.0000000
X43	25	1.0000000
X44	1	0.0097095
X44	5	1.0000000
X44	22	1.0000000
X44	26	1.0000000
X51	1	1.0000000
X51	6	1.0000000
X51	22	1.0000000
X51	23	1.0000000
X52	1	0.0000000
X52	6	1.0000000
X52	22	1.0000000
X52	24	1.0000000
X53	1	0.0172866
X53	6	1.0000000
X53	22	1.0000000
X53	25	1.0000000
X54	1	0.3908050
X54	6	1.0000000
X54	22	1.0000000
X54	26	1.0000000
X62	1	0.0000004
X62	7	1.0000000
X62	22	1.0000000
X62	24	1.0000000
X63	1	0.4461143
X63	7	1.0000000
X63	22	1.0000000

X63	25	1.0000000
X64	1	0.0962878
X64	7	1.0000000
X64	22	1.0000000
X64	26	1.0000000
X71	1	1.0000000
X71	8	1.0000000
X71	22	1.0000000
X71	23	1.0000000
X72	1	0.0000003
X72	8	1.0000000
X72	22	1.0000000
X72	24	1.0000000
X73	1	0.0045918
X73	8	1.0000000
X73	22	1.0000000
X73	25	1.0000000
X74	1	0.0230059
X74	8	1.0000000
X74	22	1.0000000
X74	26	1.0000000
X81	1	1.0000000
X81	9	1.0000000
X81	22	1.0000000
X81	23	1.0000000
X82	1	0.0000001
X82	9	1.0000000
X82	22	1.0000000
X82	24	1.0000000
X83	1	0.0881944
X83	9	1.0000000
X83	22	1.0000000
X83	25	1.0000000
X84	1	0.0102261
X84	9	1.0000000
X84	22	1.0000000
X84	26	1.0000000
X92	1	0.0016313
X92	10	1.0000000
X92	22	1.0000000
X92	24	1.0000000
X93	1	0.0608889
X93	10	1.0000000
X93	22	1.0000000
X93	25	1.0000000
X94	1	0.0122875
X94	10	1.0000000
X94	22	1.0000000
X94	26	1.0000000
X102	1	0.0000003
X102	11	1.0000000
X102	22	1.0000000
X102	24	1.0000000
X103	1	0.3939155

X103	11	1.0000000
X103	22	1.0000000
X103	25	1.0000000
X104	1	0.0809395
X104	11	1.0000000
X104	22	1.0000000
X104	26	1.0000000
X111	1	1.0000000
X111	12	1.0000000
X111	22	1.0000000
X111	23	1.0000000
X112	1	0.0000000
X112	12	1.0000000
X112	22	1.0000000
X112	24	1.0000000
X113	1	0.0381849
X113	12	1.0000000
X113	22	1.0000000
X113	25	1.0000000
X114	1	0.0100352
X114	12	1.0000000
X114	22	1.0000000
X114	26	1.0000000
X121	1	1.0000000
X121	13	1.0000000
X121	22	1.0000000
X121	23	1.0000000
X122	1	0.0000246
X122	13	1.0000000
X122	22	1.0000000
X122	24	1.0000000
X123	1	0.0532326
X123	13	1.0000000
X123	22	1.0000000
X123	25	1.0000000
X124	1	0.4369746
X124	13	1.0000000
X124	22	1.0000000
X124	26	1.0000000
X132	1	0.0000020
X132	14	1.0000000
X132	22	1.0000000
X132	24	1.0000000
X133	1	0.1116613
X133	14	1.0000000
X133	22	1.0000000
X133	25	1.0000000
X134	1	0.0182207
X134	14	1.0000000
X134	22	1.0000000
X134	26	1.0000000
X142	1	0.0000000
X142	15	1.0000000
X142	22	1.0000000

X142	24	1.0000000
X143	1	0.1126071
X143	15	1.0000000
X143	22	1.0000000
X143	25	1.0000000
X144	1	0.0330128
X144	15	1.0000000
X144	22	1.0000000
X144	26	1.0000000
X152	1	0.0000027
X152	16	1.0000000
X152	22	1.0000000
X152	24	1.0000000
X153	1	0.3694486
X153	16	1.0000000
X153	22	1.0000000
X153	25	1.0000000
X154	1	0.0324104
X154	16	1.0000000
X154	22	1.0000000
X154	26	1.0000000
X161	1	1.0000000
X161	17	1.0000000
X161	22	1.0000000
X161	23	1.0000000
X162	1	0.0000000
X162	17	1.0000000
X162	22	1.0000000
X162	24	1.0000000
X163	1	0.7269329
X163	17	1.0000000
X163	22	1.0000000
X163	25	1.0000000
X164	1	0.0887002
X164	17	1.0000000
X164	22	1.0000000
X164	26	1.0000000
X171	1	1.0000000
X171	18	1.0000000
X171	22	1.0000000
X171	23	1.0000000
X172	1	0.0000001
X172	18	1.0000000
X172	22	1.0000000
X172	24	1.0000000
X173	1	0.1214913
X173	18	1.0000000
X173	22	1.0000000
X173	25	1.0000000
X174	1	0.0795344
X174	18	1.0000000
X174	22	1.0000000
X174	26	1.0000000
X181	1	1.0000000

```
X181        19          1.0000000
X181        22          1.0000000
X181        23          1.0000000
X182        1           0.0000000
X182        19          1.0000000
X182        22          1.0000000
X182        24          1.0000000
X183        1           0.0034708
X183        19          1.0000000
X183        22          1.0000000
X183        25          1.0000000
X184        1           0.0690683
X184        19          1.0000000
X184        22          1.0000000
X184        26          1.0000000
X191        1           1.0000000
X191        20          1.0000000
X191        22          1.0000000
X191        23          1.0000000
X192        1           0.0000299
X192        20          1.0000000
X192        22          1.0000000
X192        24          1.0000000
X193        1           0.0053682
X193        20          1.0000000
X193        22          1.0000000
X193        25          1.0000000
X194        1           0.0071430
X194        20          1.0000000
X194        22          1.0000000
X194        26          1.0000000
X202        1           0.0000007
X202        21          1.0000000
X202        22          1.0000000
X202        24          1.0000000
X203        1           0.7371324
X203        21          1.0000000
X203        22          1.0000000
X203        25          1.0000000
X204        1           0.0262020
X204        21          1.0000000
X204        22          1.0000000
X204        26          1.0000000
DP1         1           0.0000000
DP1         23         -1.0000000
DP2         1           0.0000000
DP2         24         -1.0000000
DP3         1           0.0000000
DP3         25         -1.0000000
DP4         1           0.0000000
DP4         26         -1.0000000
DM1         1          -5.0000000
DM1         23          1.0000000
DM2         1         -10.0000000
```

```
    DM2        24              1.0000000
    DM3        1             -10.0000000
    DM3        25              1.0000000
    DM4        1              -5.0000000
    DM4        26              1.0000000
RHS
    RHS        2               1.0000000
    RHS        3               1.0000000
    RHS        4               1.0000000
    RHS        5               1.0000000
    RHS        6               1.0000000
    RHS        7               1.0000000
    RHS        8               1.0000000
    RHS        9               1.0000000
    RHS        10              1.0000000
    RHS        11              1.0000000
    RHS        12              1.0000000
    RHS        13              1.0000000
    RHS        14              1.0000000
    RHS        15              1.0000000
    RHS        16              1.0000000
    RHS        17              1.0000000
    RHS        18              1.0000000
    RHS        19              1.0000000
    RHS        20              1.0000000
    RHS        21              1.0000000
    RHS        22             20.0000000
    RHS        23              5.0000000
    RHS        24              6.0000000
    RHS        25              4.0000000
    RHS        26              5.0000000
ENDATA
```

```
LINDO/386 5.1 (21 DEC 92)

COPYRIGHT (C) 1984-92 LINDO SYSTEMS,  INC.  LICENSED
MATERIAL, ALL RIGHTS RESERVED. COPYING EXCEPT AS
AUTHORIZED IN LICENSE AGREEMENT IS PROHIBITED.

SINGLE USER LICENSE LDPC3-511106
STATCOM, INC.

NAME     LINDO GENERATED MPS FILE (MAX)

CANDIDATE OBJECTIVE ROW(S) IS(ARE):
1
MAX OR MIN ?
ROWS=     26 VARS=     79 NO. INTEGER VARS=      0
NONZEROES=     306 CONSTRAINT NONZ=     221(     221 ARE +- 1)
DENSITY=0.147
SMALLEST AND LARGEST ELEMENTS IN ABSOLUTE VALUE=   0.200000E-05
20.0000
NO. < :     0 NO. =:     25 NO. > :     0, OBJ=MAX, GUBS <=   20
SINGLE COLS=     4
 WARNING: PROBLEM IS POORLY SCALED.  THE UNITS
OF THE ROWS AND VARIABLES SHOULD BE CHANGED SO
THE COEFFICIENTS COVER A MUCH SMALLER RANGE.
LP OPTIMUM FOUND AT STEP     48

       OBJECTIVE FUNCTION VALUE

       1)      8.9594040

 VARIABLE        VALUE          REDUCED COST
      X12         1.000000          0.000000
      X13         0.000000          0.407697
      X14         0.000000          0.067406
      X22         0.000000          0.154993
      X23         1.000000          0.000000
      X24         0.000000          0.209065
      X31         0.000000          0.416383
      X32         0.000000          0.416412
      X33         0.000000          0.790869
      X34         1.000000          0.000000
      X41         1.000000          0.000000
      X42         0.000000          0.000029
      X43         0.000000          0.357172
      X44         0.000000          0.059359
      X51         0.000000          0.321737
      X52         0.000000          0.321766
      X53         0.000000          0.723346
      X54         1.000000          0.000000
      X62         0.000000          0.027248
      X63         1.000000          0.000000
      X64         0.000000          0.000000
      X71         1.000000          0.000000
      X72         0.000000          0.000029
```

VARIABLE	VALUE	REDUCED COST
X73	0.000000	0.414304
X74	0.000000	0.046063
X81	1.000000	0.000000
X82	0.000000	0.000029
X83	0.000000	0.330701
X84	0.000000	0.058842
X92	1.000000	0.000000
X93	0.000000	0.359609
X94	0.000000	0.058383
X102	0.000000	0.011900
X103	0.000000	0.036851
X104	1.000000	0.000000
X111	1.000000	0.000000
X112	0.000000	0.000029
X113	0.000000	0.380711
X114	0.000000	0.059033
X121	0.000000	0.367906
X122	0.000000	0.367911
X123	0.000000	0.733569
X124	1.000000	0.000000
X132	1.000000	0.000000
X133	0.000000	0.307207
X134	0.000000	0.050821
X142	1.000000	0.000000
X143	0.000000	0.306259
X144	0.000000	0.036027
X152	1.000000	0.000000
X153	0.000000	0.049420
X154	0.000000	0.036631
X161	0.000000	0.308037
X162	0.000000	0.308066
X163	1.000000	0.000000
X164	0.000000	0.288405
X171	0.000000	0.010466
X172	0.000000	0.010495
X173	0.000000	0.307870
X174	1.000000	0.000000
X181	1.000000	0.000000
X182	0.000000	0.000029
X183	0.000000	0.415425
X184	0.000000	0.000000
X191	0.000000	0.000000
X192	1.000000	0.000000
X193	0.000000	0.413527
X194	0.000000	0.061925
X202	0.000000	0.318266
X203	1.000000	0.000000
X204	0.000000	0.361103
DP1	0.000000	4.069068
DP2	0.000000	5.069039
DP3	0.000000	4.650173
DP4	0.000000	5.000000
DM1	0.000000	0.930932

```
VARIABLE        VALUE          REDUCED COST
     DM2         0.000000          4.930961
     DM3         0.000000          5.349827
     DM4         0.000000          0.000000

     ROW   SLACK OR SURPLUS     DUAL PRICES
      2)         0.000000         -0.000029
      3)         0.000000          0.154964
      4)         0.000000          0.416383
      5)         0.000000          0.000000
      6)         0.000000          0.321737
      7)         0.000000          0.027219
      8)         0.000000          0.000000
      9)         0.000000          0.000000
     10)         0.000000          0.001602
     11)         0.000000          0.011871
     12)         0.000000          0.000000
     13)         0.000000          0.367906
     14)         0.000000         -0.000027
     15)         0.000000         -0.000029
     16)         0.000000         -0.000027
     17)         0.000000          0.308037
     18)         0.000000          0.010466
     19)         0.000000          0.000000
     20)         0.000000          0.000000
     21)         0.000000          0.318237
     22)         0.000000          5.069068
     23)         0.000000         -4.069068
     24)         0.000000         -5.069039
     25)         0.000000         -4.650173
     26)         0.000000         -5.000000

NO. ITERATIONS=       48

DO RANGE(SENSITIVITY) ANALYSIS?
```

SUMMARY OF THE RESULT

Course	Soldiers	Match	CO	EL	FA	GT	SC	ST
18B	5	0	122	107	124	116	119	107
18C	6	1	123	116	118	117	122	118
18D	4	2	116	115	118	116	115	117
18E	5	0	119	113	116	119	115	114

INDIVIDUAL RECORD BY MOS

MOS	SSN	WANT	GRAD	CO	EL	FA	GT	SC	ST
18B	134-52-1940	18C	E-5	128	110	118	114	121	108
18B	392-64-5487	18E	E-5	110	106	108	112	112	109
18B	457-43-9022	18B	E-5	131	117	123	122	128	118
18B	461-13-6662	18D	E-5	124	112	108	112	113	101
18B	579-78-7410	18D	E-5	108	101	107	113	108	103
18C	150-48-2195	18B	E-6	113	98	125	110	111	97
18C	240-08-0587	18E	E-4	108	109	112	115	112	113
18C	434-19-8796	18E	E-4	139	115	128	124	130	114
18C	505-70-0327	18C	E-4	114	107	112	114	112	112
18C	519-96-7811	18D	E-4	105	109	125	110	119	109
18C	560-41-9486	18D	E-4	109	111	114	118	116	123
18D	170-60-0281	18C	E-5	138	139	138	130	138	139
18D	246-35-6948	18D	E-4	111	115	113	111	121	123
18D	553-57-6920	18C	E-4	112	109	105	110	118	116
18D	572-25-4534	18D	E-4	128	120	133	120	119	128
18E	100-62-2209	18D	E-5	115	123	127	125	112	124
18E	227-17-2548	18D	E-6	122	121	119	125	111	120
18E	270-44-7648	18E	E-6	122	124	117	128	107	123
18E	429-41-9396	18D	E-4	118	118	120	112	118	119
18E	533-88-0430	18D	E-5	123	122	123	122	122	124

INDIVIDUAL RECORD BY WANT

WANT	MOS	SSN	GRAD	CO	EL	FA	GT	SC	ST
18B	18B	457-43-9022	E-5	131	117	123	122	128	118
18B	18C	150-48-2195	E-6	113	98	125	110	111	97
18C	18B	134-52-1940	E-5	128	110	118	114	121	108
18C	18C	505-70-0327	E-4	114	107	112	114	112	112
18C	18D	170-60-0281	E-5	138	139	138	130	138	139
18C	18D	553-57-6920	E-4	112	109	105	110	118	116
18D	18B	461-13-6662	E-5	124	112	108	112	113	101
18D	18B	579-78-7410	E-5	108	101	107	113	108	103
18D	18C	519-96-7811	E-4	105	109	125	110	119	109
18D	18C	560-41-9486	E-4	109	111	114	118	116	123
18D	18D	246-35-6948	E-4	111	115	113	111	121	123
18D	18D	572-25-4534	E-4	128	120	133	120	119	128
18D	18E	100-62-2209	E-5	115	123	127	125	112	124
18D	18E	227-17-2548	E-6	122	121	119	125	111	120
18D	18E	429-41-9396	E-4	118	118	120	112	118	119
18D	18E	533-88-0430	E-5	123	122	123	122	122	124
18E	18B	392-64-5487	E-5	110	106	108	112	112	109
18E	18C	240-08-0587	E-4	108	109	112	115	112	113
18E	18C	434-19-8796	E-4	139	115	128	124	130	114
18E	18E	270-44-7648	E-6	122	124	117	128	107	123

INDIVIDUAL RECORD BY SSN

SSN	WANT	MOS	GRAD	CO	EL	FA	GT	SC	ST
100-62-2209	18D	18E	E-5	115	123	127	125	112	124
134-52-1940	18C	18B	E-5	128	110	118	114	121	108
150-48-2195	18B	18C	E-6	113	98	125	110	111	97
170-60-0281	18C	18D	E-5	138	139	138	130	138	139
227-17-2548	18D	18E	E-6	122	121	119	125	111	120
240-08-0587	18E	18C	E-4	108	109	112	115	112	113
246-35-6948	18D	18D	E-4	111	115	113	111	121	123
270-44-7648	18E	18E	E-6	122	124	117	128	107	123
392-64-5487	18E	18B	E-5	110	106	108	112	112	109
429-41-9396	18D	18E	E-4	118	118	120	112	118	119
434-19-8796	18E	18C	E-4	139	115	128	124	130	114
457-43-9022	18B	18B	E-5	131	117	123	122	128	118
461-13-6662	18D	18B	E-5	124	112	108	112	113	101
505-70-0327	18C	18C	E-4	114	107	112	114	112	112
519-96-7811	18D	18C	E-4	105	109	125	110	119	109
533-88-0430	18D	18E	E-5	123	122	123	122	122	124
553-57-6920	18C	18D	E-4	112	109	105	110	118	116
560-41-9486	18D	18C	E-4	109	111	114	118	116	123
572-25-4534	18D	18D	E-4	128	120	133	120	119	128
579-78-7410	18D	18B	E-5	108	101	107	113	108	103

931109